Deep Exploration via Stochastic Gradient Langevin Dynamic in Deep Reinforcement Learning

Anonymous Authors[1]

Abstract

This document provides a basic paper template and submission guidelines. Abstracts must be a single paragraph, ideally between 4–6 sentences long. Gross violations will trigger corrections at the camera-ready phase.

1. Introduction

The application of deep learning in reinforcement learning has brought a lot of research progress (Mnih et al., 2015; Silver et al., 2016; OpenAI, 2018). Neural networks can fit arbitrary functions and have many extended forms on different tasks, which enabling reinforcement learning to solve various tasks, such as CNN for pixel input and RNN for non-Markov tasks(Mnih et al., 2015; Bakker, 2002). The latest generation of deep learning frameworks has convenient automatic derivation, integrate various optimizers (Paszke et al., 2017; Chen et al., 2015; Abadi et al., 2016). As in supervised learning, the "loss function + SGD-base optimizer" approach becomes the general way of using neural networks in RL, both for Q-learning or policy gradient (Lillicrap et al., 2015; Mnih et al., 2015; Schulman et al., 2017).

However, in more difficult tasks, especially those with continuous state-action space, or sparse/deceptive rewards, more effective exploration strategies become the key to solving the problem (Plappert et al., 2017; Colas et al., 2018). In the sparse reward task, except for a few specific events, the agent can only get 0 rewards, which will require the agent to effectively traverse the policy space without any incentives (Houthooft et al., 2016). In the deceptive reward task, due to the action penalty in the reward, the agent will receive negative feedback when exploring, which leads the agent to abandon the exploration and maintain the "zero action" (Lehman & Stanley, 2011; Conti et al., 2018b). This situation requires the agent to continue to explore the unknown part of the state action space. In the opposite case, if the task has a well-defined reward, which can guide the agent to learn the optimal strategy, too much exploration will hurt the sample efficiency (Riquelme et al., 2018). The trade-off between exploration and exploitation is an important research issue for reinforcement learning.

There has been a lot of research work on the agent's exploration strategy, including some naive random search methods (action space or parameter space) (Plappert et al., 2017; Lillicrap et al., 2015), count-based methods (Tang et al., 2017; Bellemare et al., 2016), evolutionary strategy (Khadka & Tumer, 2018; Conti et al., 2018a) and so on. There is a class of methods based on Thomson's sampling (Thompson, 1933), the core idea of which is to make decisions using samples generated from the posterior distribution of the value model on the current data. Thompson sampling has an "Instance-Independent Regret bounds" (Russo et al., 2018) and has proven to be effective in a variety of deep learning-based reinforcement learning scenarios (Osband et al., 2016; Houthooft et al., 2016; Henderson et al., 2017; Azizzadenesheli et al., 2018).

However, for an algorithm that uses a neural network as a function of value estimation, it is difficult to directly calculate the posterior distribution. In response to this, there are a series of works to achieve approximate a posteriori sampling using various approximation methods, including dropout (Henderson et al., 2017), variational inference (Houthooft et al., 2016), Bootstrap (Osband et al., 2016), and so on. However, the posterior distributions followed by these methods are not accurate, and there may be problems such as lack of intrinsic motivation and computational cost. (Osband et al., 2018)

It is noted that the training process of optimizing the value estimation function of Bellman error has many similarities with the supervised learning. We use a stochastic gradient based Stochastic Gradient Langevin Dynamics (SGLD) to achieve posterior sampling (Welling & Teh, 2011). This work uses the Deep Deterministic Policy Gradient (DDPG) algorithm as the baseline algorithm (Lillicrap et al., 2015). We use the SGLD sampler instead of the value estimate function's optimizer to continuously generate a posteriori samples of the current observations during the training step,

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

and use these samples to generate policy gradient optimization actors for exploration during the rollout phase. Further, we consider the parts of the original algorithm that are not coordinated with the posterior sampling and adjust them to obtain a more uniform sample. This is a more accurate way to sample from the posterior and introduces little additional computational cost.Further, we adjusted the portion of the baseline algorithm that is not coordinated with the posterior sampling to achieve a more consistent posteriori sampling. This is a more accurate way to sample from the posterior and introduces little additional computational cost.

In this work, we validated the performance of our exploration method with continuous MountainCar, Sparse HalfCheetah, and HalfCheetah as deceptive, sparse, and well-defined tasks, and compared it with Action noise, Parameter noise. In general, SGLD-based approaches can solve a wider variety of tasks with uniform settings and get better results in well-defined tasks. We summarize the three main contributions of this work:

- It is the first time to propose the use of the SGLD method to achieve deep exploration in RL which use DNN as a value estimate function.
- The method of using SGLD sampler to do Thomson sampling in DDPG algorithm is proposed. No hyperparameter is introduced, and there is almost no additional computational overhead.
- This work proves through experiments that the method of this paper can solve the three different tasks of sparse reward, deception reward and good definition by using uniform settings.

This paper is organized as follows: In the section 2, we summarize some existing exploration methods and explain the relationship between our work and them. The third section briefly introduces the DDPG algorithm and the principle of SGLD, and gives a detailed description of how we apply SGLD in DDPG. In Section 4 we conducted experiments on several representative tasks and analyzed why our methods are effective. Finally, we summarize this work in section 5 and point out our opinions on further research work.

2. Related works

Reinforcement learning algorithms use neural networks as Q-value functions or policy functions achieve remarkable results in the continuous state-action space tasks . In order to solve more difficult tasks such as sparse or deceptive reward tasks, more effective exploration strategies have become a key point.

Some naive random search algorithms are generally used as default exploration strategies by various RL algorithms. Such as the ϵ-greed method which takes non-optimal action according to probability $1-\epsilon$ (Mnih et al., 2015), or directly superimposes noise (correlation or not) in action space (Lillicrap et al., 2015). Another approach is to inject noise in the parameter space of the Q-value function of the policy function (Plappert et al., 2017). In this situation, the agent will take the same action in the face of the same state within an episode, which leads to more consistent exploration behavior. However, such methods do not explicitly suggest how to use existing data to guide exploration, resulting in low sample efficiency.

The Thompson sampling method, as a long-history exploration strategy, has recently returned to the perspective of deep reinforcement learning researchers. This method draws a sample from the posterior distribution of the Q-value function and makes decisions based on the random sample rather than an optimal estimation. Since it is difficult to directly calculate the posterior distribution of the neural network, the key point in the work of using Thompson sampling in reinforcement learning is the approach to achieve posterior sampling. For example, BDQN (Osband et al., 2016) and prior-BDQN (Osband et al., 2018) uses a multi-head network trained in bootstrapping way and treats the set of output as sampled results. VIME (Houthooft et al., 2016) and Stein Variational Policy Gradient (Liu et al., 2017) use the reparameter trick and the stein gradient approach to implement variational inference as an approximation of the posterior distribution. These methods lack intrinsic motivation in the zero reward environment (Except prior-BDQN) and introduce considerable computational overhead.

In this work, we use the SGLD to draw samples from the posterior distribution during training the value estimation function. There are a number of variants of SGLD, such as pre-condiction (Li et al., 2016), SGHMC (Chen et al., 2014), SGFS(Ahn et al., 2012), etc. The pre-condiction SGLD can be seen as a variant of RMSprop and has as almost equal number of operations, and been used as the sampler in this work. Because SGLD can be seen as accumulatng Gaussian noise during the optimization process, our method has similarities with parameter noise. The difference is that we add accumulated noise to the critic by SGLD, while they only add one-time noise to the actor when rollout samples. On the other hand, in prior-BDQN, they prepare a set of "prior function" in advance then select some of them to add on Q-value function during training, while in our algorithm there only have a single-head network with accumulated noise so that our method is more computational efficient.

3. Method

3.1. Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient is an off-policy actor-critic algorithm that uses a replay buffer and two target networks (Lillicrap et al., 2015). In each training cycle, the agent rollouts some transition datas from the environment E and stores them into the replay buffer D. Then calculate the Bellman Error by equation (1) with the target network Q' and μ' and optimize the parameters θ^Q of the critic Q in several training steps. And optimize the parameters θ^μ of the actor μ by the policy gredient estimated by the critic. The target network is soft updated at the end of each training step. The details of the algorithm are described in Alg 1.

$$L^Q = \mathbb{E}_{s_t\sim\rho^{\tilde{\mu}},a_t\sim\tilde{\mu},r_t\sim E}[(Q(s_t,a_t)-y_t)^2] \quad \text{where} \quad y_t = Q'(s_{t+1},\mu'(s_{t+1}))+r_t \tag{1}$$

Since DDPG is an off policy algorithm, the rollout actor and the learning actor do not need to be the same, the exploration strategy $f(\cdot):\mathbb{M}\to\mathbb{M}$, which $\mathbb{M}$ is the space of policy, commonly transforms the learning actor μ into a rollout actor $\tilde{\mu}$ to achieve exploration.

Algorithm 1 Deep Deterministic Policy Gradient

Input: environment E, critic $Q(s,a|\theta^Q)$, actor $\mu(s|\theta^\mu)$,
exploration strategy $\tilde{\mu}\leftarrow f(\mu)$
Initialize replay buffer $D=\emptyset$.
Initialize target network $Q'=Q$, $\mu'=\mu$.
for 1 **to** cycle_number **do**
  Apply exploration strategy $\tilde{\mu}\leftarrow f(\mu)$
  Rollout data d_t from E by $\tilde{\mu}$ and $D\leftarrow D\cup d_t$
  for 1 **to** train_steps **do**
    Sample data batch $\{d_t=(s_t,a_t,r_t)\}$ from D
    $L^Q=\sum(Q'(s_{t+1},\mu'(s_{t+1}))+r_t-Q(s_t,a_t))^2$
    $L^\mu=-\sum Q(\mu(s_t))$
    $\theta^Q\leftarrow\theta^Q-\alpha^Q\cdot\nabla_{\theta^Q}L^Q$
    $\theta^\mu\leftarrow\theta^\mu-\alpha^\mu\cdot\nabla_{\theta^\mu}L^\mu$
    Soft update target network Q' and μ'
  end for
end for

3.2. Stochastic Gradient Langevin Dynamic

Stochastic Gradient Langevin Dynamic algorithm is an algorithm that can perform MCMC sampling on a large dataset in stochastic gradient way (Welling & Teh, 2011). For a paramter vector θ with a prior distribution $p(\theta)$,to draw a sample chain $\{\theta_1,\theta_2,\cdots\}$ follow $p(\theta|D)$ where $D=\{d_i\}_{i=1}^N$, the parameters should be updated as follow:

$$\Delta\theta = \frac{\epsilon}{2}\frac{N}{n}\sum_{i=1}^{n}\nabla_\theta\log p(d_i|\theta)+\frac{\epsilon}{2}\nabla_\theta\log p(\theta)+\mathcal{N}(0,\epsilon) \tag{2}$$

Where ϵ is the learning rate, n is the size of one mini-batch.

The basic SGLD algorithm updates all parameters with the same step size, which leads to slow mixing rate. In practical, we use preconditioned SGLD (pSGLD) instead, in which the step size is scaled by preconditioner G_t as follow:

$$\Delta\theta = \frac{\epsilon}{2}\frac{N}{n}G_t\sum_{i=1}^{n}\nabla_\theta\log p(d_i|\theta)+\frac{\epsilon}{2}G_t\nabla_\theta\log p(\theta)+\mathcal{N}(0,\epsilon G_t) \tag{3}$$

Where G_t is updated as the mean square term in RMSprop (Tieleman & Hinton, 2012).

To visualize the ability of pSGLD to characterization the uncertainty of posterior distribution when fitting data set, we trained a neural network with architecture of "20-ReLU-20-ReLU-1" by pSGLD on three toy datasets, left: $y=0$, mid: $y=-x^2$, right: $y=x^3-x$, and the results are shown in the figure 1. The dark curve is the average of the curve clusters, and the light areas represent the standard deviation of the curve clusters. The results show that the curve samples of pSGLD are concentrated near the data points, while diverge in both positive and negative directions in areas far from the data, even on the all zero data set.

3.3. SGLD in DDPG

In baseline DDPG algorithm, the parameters of critic network θ^Q is updated by an Adam optimizer to minimize the Bellman error L^Q. To replace the Adam optimizer by pSGLD sampler with minimal changes, we suppose that the likelihood term is $p(d_i|\theta^Q)\sim\exp(-L^Q)$ to match the Bellman error, and prior term is $p(\theta^Q)\sim\mathcal{N}(0,\sigma^2)$ to match the L2 regularization on critic network. Then the critic will updated as follow :

$$\Delta\theta^Q = -\frac{\epsilon}{2}\frac{N}{n}G_t\sum_{i=1}^{n}\nabla_\theta L^Q-\frac{\epsilon}{2}G_t\frac{\theta^Q}{\sigma^2}+\mathcal{N}(0,\epsilon G_t) \tag{4}$$

It is worth noting that, unlike supervised learning, the size of data sets N in reinforcement learning is increasing as the training process increases. If the learning rate ϵ remains constant, the gradient of the likelihood term will continue to grow, resulting in an unstable training process. In order to keep the training process stable, we implement learning rate decay as opposed to data set size growth: $\epsilon_t=\frac{n}{N}\epsilon_0$. Now the equation (4) turns into:

$$\Delta\theta^Q = -\frac{\epsilon_0}{2}G_t\sum_{i=1}^{n}\nabla_\theta L^Q-\frac{\epsilon_0}{2}\frac{n}{N}G_t\frac{\theta^Q}{\sigma^2}+\mathcal{N}(0,\epsilon_0\frac{n}{N}G_t) \tag{5}$$

The observed data is increasing as the learning process progresses, which leads to the effect of the prior term and

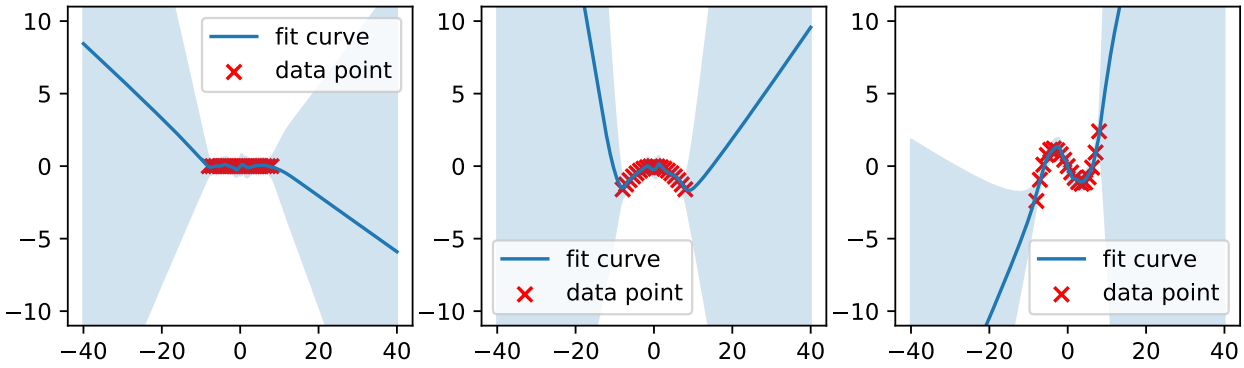

Figure 1. The results of curve fitting by using pSGLD on three toy data sets.

the noise term becomes weaker than the likelihood term. This is consistent with our common sense: the more data is observed, the more the model's distribution is concentrated toward the maximum likelihood estimate.

3.4. Further follow posterior distribution

In original DDPG algorithm, the target networks are the τ-moving average of parameters of actor and critic. After each gradient descent step on actor and critic ,the target networks are "soft updated" by equation (6).

$$\begin{aligned} \theta^{Q'} &= (1-\tau)\theta^{Q'} + \tau\theta^{Q} \\ \theta^{\mu'} &= (1-\tau)\theta^{\mu'} + \tau\theta^{\mu} \end{aligned} \tag{6}$$

The optimized object for the critic in one cycle is minimizing Bellman error in equation (1), which is associated with both data in replay buffer and the target network. Constantly updating the target during the training step will make the process of sampling from the posterior distribution inconsistently. In order to sample critic from posterior distribution more strictly, we remain the target unchanged in one cycle and do "hard update",directly copy parameters of actor and critic to target networks. Under this setting, the expectation policy gradient will be intergrated over both $s_t \sim \rho$ and $\theta^Q \sim p(\theta^Q|D)$ as equation (7).

$$\nabla_{\theta^\mu}\mathbb{E}[L^\mu|D] = \mathbb{E}_{s_t\sim\rho,\theta^Q\sim p(\theta^Q|D)}[\nabla_{\theta^\mu}L^\mu] \tag{7}$$

It is proven to be the correct form to estimate policy gradient with critics sampled from posterior distribution in (Henderson et al., 2017).

4. Results and Discussion

4.1. Experiment setting

We choose MontainCar_continous(MC) as a deceptive reward task, SparseHalfCheetah(SHC) as a sparse reward task,and HalfCheetah(HC) as well defined task.

We use Ubuntu16.04 on XX CPU+XX GPU. We use gym XX + mujoco 1.5.1

4.2. Sample critic with uncertainty

We rollout some transitions in MC, and plot the state point in "position-velocity". Then we train a critic network with these data by Adam optimizer and SGLD sampler. In both cases we start with an random initialized network and pretrain the critic network for XX steps to make sure convergence. After preparation, we further optimize the network for XX steps, and store the copy of critic network after each optimize step as one sample. BalaBala....

We can see SGLD can sample critic which have high uncertainty on the periphery state domain.

4.3. Exploration in sparse or deceptive tasks

We compared the effects of various exploration methods on the MC and SHC tasks.

4.4. Exploration in well defined tasks

We compared the effects of various exploration methods on the HC task.

5. Conclusion

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Ahn, S., Korattikara, A., and Welling, M. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv*

preprint arXiv:1206.6380, 2012.

Azizzadenesheli, K., Brunskill, E., and Anandkumar, A. Efficient exploration through bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*, 2018.

Bakker, B. Reinforcement learning with long short-term memory. In *Advances in neural information processing systems*, pp. 1475–1482, 2002.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Chen, T., Fox, E., and Guestrin, C. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, pp. 1683–1691, 2014.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Colas, C., Sigaud, O., and Oudeyer, P.-Y. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. *arXiv preprint arXiv:1802.05054*, 2018.

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K., and Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pp. 5032–5043, 2018a.

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K., and Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pp. 5032–5043, 2018b.

Henderson, P., Doan, T., Islam, R., and Meger, D. Bayesian policy gradients via alpha divergence dropout inference. *arXiv preprint arXiv:1712.02037*, 2017.

Henderson, P., Doan, T., Islam, R., and Meger, D. Bayesian policy gradients via alpha divergence dropout inference. *NIPS Bayesian Deep Learning Workshop*, 2017.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Khadka, S. and Tumer, K. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1196–1208, 2018.

Lehman, J. and Stanley, K. O. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

Li, C., Chen, C., Carlson, D. E., and Carin, L. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *AAAI*, volume 2, pp. 4, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., and Wierstra, D. Continuous control with deep reinforcement learning. *Computer Science*, 8(6):A187, 2015.

Liu, Y., Ramachandran, P., Liu, Q., and Peng, J. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., and Ostrovski, G. J. N. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015. ISSN 1476-4687.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pp. 4026–4034, 2016.

Osband, I., Aslanides, J., and Cassirer, A. Randomized prior functions for deep reinforcement learning. *arXiv preprint arXiv:1806.03335*, 2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Riquelme, C., Tucker, G., and Snoek, J. Deep bayesian bandits showdown. In *International Conference on Learning Representations*, 2018.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z., et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. J. n. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016. ISSN 1476-4687.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2753–2762, 2017.

Thompson, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.