A Linear Online Guided Policy Search Algorithm

Biao Sun[1,2], Fangzhou Xiong[2,3], Zhiyong Liu[2,3,4,5](✉),
Xu Yang[2], and Hong Qiao[1,2,3,4,5]

[1] University of Science and Technology Beijing, Beijing 100083, China
[2] The State Key Lab of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Science, Beijing 100190, China
zhiyong.liu@ia.ac.cn
[3] School of Computer and Control, University of Chinese Academy of Sciences (UCAS), Beijing 100049, China
[4] CAS Centre for Excellence in Brain Science and Intelligence Technology (CEBSIT), Shanghai 200031, China
[5] Cloud Computing Center, Chinese Academy of Sciences, DongGuan 523808, Guandong, China

Abstract. In reinforcement learning (RL), the guided policy search (GPS), a variant of policy search method, can encode the policy directly as well as search for optimal solutions in the policy space. Even though this algorithm is provided with asymptotic local convergence guarantees, it can not work in a online way for conducting tasks in complex environments since it is trained with a batch manner which requires that all of the training samples should be given at the same time. In this paper, we propose an online version for GPS algorithm, which can learn policies incrementally without complete knowledge of initial positions for training. The experiments witness its efficacy on handling sequentially arriving training samples in a peg insertion task.

Keywords: Reinforcement learning · Policy search · Online learning

1 Introduction

Reinforcement learning (RL) provides robotics a framework with a set of tools for the design of sophisticated and hard-to-engineer behaviors to interact with realistic world. It enables a robot to autonomously discover an optimal behavior through trial-and-error interactions with its environment. As an important field in reinforcement learning, policy search methods have been used in robotics for a wide range of tasks, such as manipulation [1], grasping [2], and locomotion [3], which scale RL into high dimensional continuous action spaces by using parameterized policies to avoid bootstrapping introduced by traditional value-function approximation. However, direct policy search usually requiring numerous samples to find optimal policy which is impractical for robot learning [4].

Guided policy search (GPS) tackles the issue of sample efficiency by introducing trajectory optimization to guide the policy search away from poor local optima [5]. This approach commonly uses trajectory-centric method to generate suitable samples at all training conditions to guide the learning process and train complex, high-dimensional policies [6–8]. Nevertheless, GPS method cannot cope with incremental data processing due to its framework.

The GPS algorithm would cause a problem that it cannot learn continuously based on previous policies when the environment changes. In order to learn a new condition, for example, a task of peg insertion with a new initial position in our experiment, the procedure for optimizing new condition should be added to all steps. As discussed in the next section, new condition's learning process will directly affect global policy optimization in the outer loop, which could obviously have an influence on other local policies indirectly through linear global policy $\pi_\theta^{'}$. These mutual impacts could be considered as the intrinsic characteristics of GPS from the view of initial conditions for training policies. Therefore, traditional GPS needs to learn all conditions from scratch, which is however hard-to-satisfied in some real applications. To alleviate this drawback, there is a great need to learn policies in an incremental way instead of the strict requirements with acquiring all initial conditions together.

The issue of online learning in GPS with multiple initial conditions can be taken as a part of lifelong learning since its learning never ends as new condition appears continuously. Lifelong learning has been explored for reinforcement learning [9]. Recently, an efficient policy gradient method for lifelong learning has been proposed [10]. In this paper, we aim to optimize GPS with an online learning form.

The main contributions of this paper are twofold, with the first to propose a novel framework for online GPS, and the second to give an effective algorithm to implement the idea. The proposed algorithm can utilize incremental information to search in policy parameter space in the process of interacting with the environment, which makes robot able to adjust to the changed conditions especially in industrial environments. Section 2 gives a brief review on related works, and Sect. 3 proposes the detailed algorithm. Following some preliminary experimental illustrations in Sects. 4, and 5 concludes this paper.

2 Background and Related Works

Reinforcement learning seeks to find a policy π to control an agent in a stochastic environment to finish some specific tasks. Instead to maximize the total rewards, RL often solves a optimal policy by minimizing the total costs under the policy trajectory distribution, given by

$$J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\pi_\theta}[\ell(x_t, u_t)]. \tag{1}$$

where π_θ stands for the policy distribution and $\ell(x_t, u_t)$ is the cost in step t.

The principle of the GPS is to use a series of special controllers to optimize π_θ. Since the special controllers generate guiding samples that guide policy search to the regions of high rewards, the GPS can efficiently train a deep neural network with fewer samples than direct policy search [2]. The expected cost minimizations can be rewritten as the following constrained problem,

$$\min_{p,\pi_\theta} \mathbb{E}_p[\ell(\tau)] \; s.t. \; p(u_t|x_t) = \pi_\theta(u_t|x_t) \; \forall x_t, u_t, t. \tag{2}$$

This optimization can be decomposed into two loops: the inner loop to optimize local policies and the outer loop to minimize the distance between global policy and each local policy. The algorithm is summarized in Algorithm 1. The inner loop optimizes the local policies respectively as follows:

$$p_i \leftarrow \operatorname*{argmin}_{p_i} \mathbb{E}_{p_i(\tau)}[\sum_{t=1}^{T} \ell(x_t, u_t)] \; s.t. \; D_{KL}(p_i(\tau)||\pi_\theta^{'}(\tau)) \leq \epsilon. \tag{3}$$

This subproblem can be solve by using local RL methods such as iterative linear-Gaussian regulator (iLQG) or path integral method [11].

The outer loop optimizes global policy to mimic each local policy by minimizing the KL divergence between them, which is given by

$$\min_{\theta} \sum_{m=1}^{M} \sum_{t=1}^{T} \mathbb{E}_{p_m(x_t,m)}[\mathbb{D}_{KL}(\pi_\theta(u_t|x_{t,m})||p_m(u_t|x_{t,m}))]. \tag{4}$$

where the number of local policy is M and each trajectory has T steps.

Algorithm 1. GPS contains two loops

```
1: for optimizing iteration to make pegging successfully do
2:   for position $i \in \{0, ..., M\}$ do
3:     c-step:$p_i \leftarrow \operatorname{argmin}_{p_i} \mathbb{E}_{p_i(\tau)}[\sum_{t=1}^{T} \ell(x_t, u_t)]$
4:             $s.t. \; D_{KL}(p_i(\tau)||\pi_\theta^{'}(\tau)) \leq \epsilon$
5:   end for
6:   s-step:$\pi_\theta \leftarrow \operatorname{argmin}_\theta \sum_{t,i,j} \mathbb{D}_{KL}(\pi_{\theta(u_t|x_{t,i,j})}||p_i(u_t|x_{t,i,j}))$
7:         (via supervised learning)
8: end for
```

In general, the optimization of each local policy p_i depends on the last optimized global policy $\pi_\theta^{'}$ as shown in the inner loop, which is a time-varying linear Gaussian function generated by the global policy π_θ. In addition, the optimization for global policy π_θ depends on the samples generated by all local polices as shown in the outer loop. The global policy will be guided to find optimum along with the local policies optimized to optimal values.

3 The Proposed Method

3.1 Linear Online Framework for Guided Policy Search

In this section, we propose an online framework for GPS. The basic idea is that learning at a new condition should separate the interaction effects of the global policy and local policies optimization, because that it would influence both the inner and outer loops. Thus the global policy should be asynchronously learned from each single local policy. Specifically, in the inner loop we optimize a single local policy directly as follows:

$$p \leftarrow \underset{p}{\operatorname{argmin}}\, \mathbb{E}_{p_\tau} \sum_{t=1}^{T} \ell(x_t, u_t). \tag{5}$$

And in the outer loop, we optimize the global policy with samples generated by each local polices. In order to learn continuously, the global policy should keep remembering the previously learned policy. Therefore, the previously learned policy should be considered to join into optimization.

The framework works like Fig. 1. The global policy learns by combining the policies learned at current condition, we denote this policy as p_{cur} and the previously learned policy as p_{pre}. The previous policies would have an influence in the learning process of the global policy all the time. Hence, the global policy could always remember previous policies regardless of how many new conditions it learns continuously.

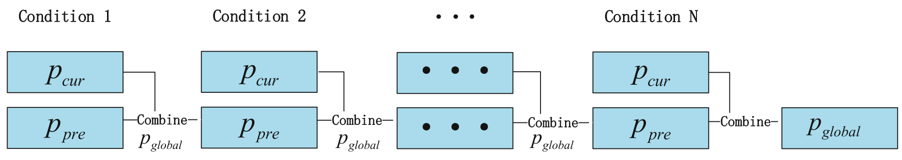

Fig. 1. The framework of online GPS.

However, the main problem is that it is hard to directly combine the global policy and the local policies, because the form of global policy would be enough complicated to represent numerous different local policies, such as neural network with multiple layers. In general, the local policy would be designed like a linear form as simple as possible to complete tasks easily and quickly.

3.2 Online Linear Guided Policy Search

In this subsection, a linear online guided policy search (LOLGPS) is proposed based on the online GPS framework shown in the Fig. 1.

Algorithm 2 summarizes our method. The inner loop, which represents learning to complete task at one condition, is the optimization for the local policies. The outer loop, which represents learning at the conditions one by one, is the optimization of the global policy. In the inner loop, we construct sample dataset D_i by running acting policy, notes as p_{act}. It is a joint controller combining the previous learnt policy into current condition, and the form is given as follows,

$$p_{act} = \alpha p_{pre} + \beta p_{cur}, \tag{6}$$

Algorithm 2. Linear online guided policy search (LOLGPS)

```
1: Initialize: $p_{pre} \leftarrow 0$
2: (outer loop)
3: for position $m = 0$ to $M$ do
4:     $p_{cur} \leftarrow$ arbitrary policy
5:     (inner loop)
6:     repeat
7:         if first iteration then
8:             Generate samples $D_i$ by running controller $p_{act} = p_{cur}$
9:         else
10:            Generate samples $D_i$ by running jointly controller $p_{act} = \alpha p_{pre} + \beta p_{cur}$
11:        end if
12:        Fit linear-Gaussian dynamics $p_i(x_{t+1}|x_t, u_t)$
13:        $p_{cur} \leftarrow \text{argmin}_p \mathbb{E}_{p(\tau)} \sum_{i=i}^{\tau} l(x_t, u_t) \;\; s.t. \; D_{kl}(p_{act}||p_{pre})$
14:        $p_{pre} \leftarrow$ use GMM to fit previous $\pi_\theta$
15:    until get enough successful samples for specific task as $D_g$
16:    $\pi_\theta \leftarrow \text{argmin}_\theta D_{kl}(\pi_\theta||p_{act})$ by using $D_g$
17: end for
```

where α and β are parameters that can be constant or dynamic and control how much information of previous learned policies will be merge into the current policy. we constrain that $\alpha + \beta = 1$. For dynamic manner, we increase α with 0.005 and decrease β with 0.005 in each iteration. The previous policy p_{pre} is a linear Gaussian representation of the learned global policy $\pi_\theta^{'}$. This loop uses iterative Linear-Quadratic Regulator (iLQR) method [8] to optimize gradually to complete task at the current condition. When using the samples affected by p_{pre}, the optimization of p_{cur} is constrained to be close to the previous policies. In this context, the equation 5 can be rewritten in step 10 as follows

$$p_{cur} \leftarrow \underset{p}{\text{argmin}} \, \mathbb{E}_{p(\tau)} \sum_{i=i}^{\tau} l(x_t, u_t) \quad s.t. \; D_{kl}(p_{cur}||p_{pre}) \leq \epsilon. \tag{7}$$

In the outer loop, we optimize global policy to mimic the local policies in current condition. To train the global policy, the samples would be collected only from the successful trajectories (line 12), as those samples that are not in successful trajectories tend to direct the global policy to a bad local optimum. Since samples are generated from the successful trajectories, the global policy would extract the representation of current policy whose form may be quite different from optimized local current policy. Using samples generated jointly by local policies and previous policy, the global policy optimization is constrained to be close to p_{act}. It is already known that $p_{act} = \alpha p_{pre} + \beta p_{cur}$ and the optimization of p_{cur} is constrained close to p_{pre}. Therefore, the global optimization also depends on p_{pre}. In this case, the step 13 can be rewritten as

$$\pi_\theta \leftarrow \underset{\theta}{\text{argmin}} \, D_{KL}(\pi_\theta||p_{cur}) + D_{KL}(\pi_\theta||p_{pre}). \tag{8}$$

Thus, the proposed method could remember historic policies while learning at new conditions continuously.

4 Experimental Illustration

4.1 Experiment Setting

A series of experiments are conducted to evaluate the proposed algorithm. The task involves a peg insertion manipulation which requires controlling a 7 DoF 3D arm to insert a tight-fitting peg into a hole. The environment simulator uses the same one in [8].

In order to generate sufficient trajectories with 100 steps, we execute the linear Gaussian controller 5 times at each initial position. For the proposed method, samples are generated from 9 optimized phases once the local policy is capable to finish pegging insertion (Fig. 2).

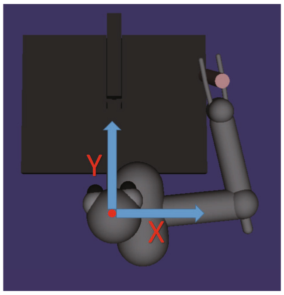

Fig. 2. Peg insertion.

Moreover, the policies are represented by a fully connected neural network with 5 layers and 42 units in each hidden layer.

The cost function is given by [2]

$$\ell(x_t, u_t) = \frac{1}{2} w_u \left|\left|u_t\right|\right|^2 + w_p \ell_{12}(p_{x_t} - p^*), \tag{9}$$

where u_t is the action of robot, p_{x_t} is the position of end effector for state x_t, and the norm $\ell_{12}(z)$ is calculated by $\frac{1}{2}\left|\left|z\right|\right|^2 + \sqrt{\gamma + z^2}$. This cost function consists of two parts, with the first one weighted by w_u to encourage the less acting energy, and the other weighted by w_p to encourage the peg to reach the hole precisely.

In addition to the cost-function values, the performances of the algorithms are also evaluated by the resulted distance between end-effector positions and target positions. In the following experiments, if the distance is smaller than a base line which is around 0.06, the task is considered to be successful.

4.2 Experimental Results and Discussion

We have carried out a set of experiments to evaluate the efficacy of the proposed method. The experiment is carried out to evaluate the performance of LOLGPS.

Experiment on LOLGPS. The proposed LOLGPS method is evaluated with different number of initial information in an incremental way. In the training phase, we first set 4 initial positions as illustrated by the blue points in Fig. 3, and randomly select two positions from the first and second training areas, respectively. In the testing phase, we randomly select 50 positions in each testing area to construct out testing sets.

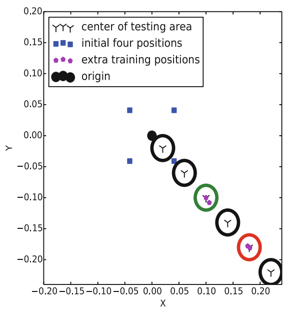

Fig. 3. Illustration of the training and testing positions, where all circles represent testing area. The green and red circles indicate the first and second extra training area, respectively.

Figure 4 illustrates the experimental results. Both the cost values and resulted distances are presented to evaluate the algorithm. The horizontal axis represents the distance between the origin and the center of each testing area. The vertical axis on the left shows the average distance between the bottom of the peg and the hole. The vertical axis on the right represents the total costs corresponding to the cost-function we used. Notes that distance_train0, distance_train1 and distance_train2 stand for the distance to target with different training areas, i.e., initial area, the first and second training area, respectively, while cost_train0, cost_train1 and cost_train2 denote corresponding cost-function values. First, it is observed that both of the resulted distance to target and the cost values increase along with the increment of testing distance. It implies that the performance becomes worse when the testing position deviates from the originally learned one. Second, it shows that once the agent has incrementally learned at new training positions, it can not only finish the tasks around the new positions, but also can still remember learned policies around the previously positions (see for instance distance_train2 and cost_train2 in Fig. 4). In other words, the proposed LOLGPS algorithm shows an ability of learning incrementally in its working environment.

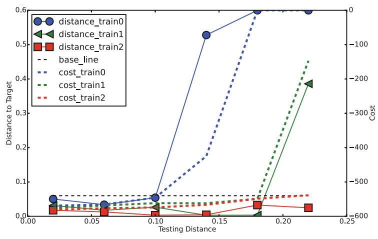

Fig. 4. Results for LOLGPS algorithm with different amounts training positions.

5 Conclusions and Future Works

In this paper, we have proposed a novel framework of GPS from the view of online learning, which enables the agent to learn policies continuously. Particularly, a LOLGPS method with linear representation has been evaluated to finish peg insertion task on simulated environment. It has been shown that the agent with proposed LOLGPS method has a potential ability of resistance to forget as given conditions appear sequentially along with interacting with environment.

Since we have separated the learning process of different condition, it is natural to think that whether the policy learning in current conditions can benefit from the learning results in other conditions. In the future work, we will pay more attention to analyze the relationship between conditions and hope to transfer the previously learned policies into current learning process so that can reduce the number of roll-out.

Acknowledgments. This work is partly supported by NSFC grants 61375005, U1613213, 61702516, 61210009, MOST grants 2015BAK35B00, 2015BAK35B01, Guangdong Science and Technology Department grant 2016B090910001.

References

1. Kalakrishnan, M., Righetti, L., Pastor, P., Schaal, S.: Learning force control policies for compliant robotic manipulation. In: Proceedings of the 29th International Conference on Machine Learning (2012)
2. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. **17**(39), 1–40 (2016)
3. Endo, G., Morimoto, J., Matsubara, T., Nakanishi, J., Cheng, G.: Learning CPG-based biped locomotion with a policy gradient method: application to a humanoid robot. Int. J. Robot. Res. **27**(2), 213–228 (2008)
4. Deisenroth, M.P., Neumann, G., Peters, J., et al.: A survey on policy search for robotics. Found. Trends Robot. **2**(1–2), 1–142 (2013)
5. Levine, S., Koltun, V.: Guided policy search. In: Proceedings of the 30th International Conference on Machine Learning, pp. 1–9 (2013)
6. Levine, S., Abbeel, P.: Learning neural network policies with guided policy search under unknown dynamics. In: Advances in Neural Information Processing Systems, pp. 1071–1079 (2014)
7. Levine, S., Koltun, V.: Variational policy search via trajectory optimization. In: Advances in Neural Information Processing Systems, pp. 207–215 (2013)
8. Montgomery, W.H., Levine, S.: Guided policy search via approximate mirror descent. In: Advances in Neural Information Processing Systems, pp. 4008–4016 (2016)
9. Sutton, R.S., Koop, A., Silver, D.: On the role of tracking in stationary environments. In: Proceedings of the 24th international conference on Machine learning, pp. 871–878 (2007)
10. Ruvolo, P., Eaton, E.: ELLA: An efficient lifelong learning algorithm. In: Proceedings of the 30th International Conference on Machine Learning, pp. 507–515 (2013)
11. Chebotar, Y., Kalakrishnan, M., Yahya, A., Li, A., Schaal, S., Levine, S.: Path integral guided policy search. In: International Conference on Robotics and Automation, pp. 3381–3388 (2017)