Artif Life Robotics (2016) 21:125–131
DOI 10.1007/s10015-015-0260-7

ORIGINAL ARTICLE

EM-based policy hyper parameter exploration: application to standing and balancing of a two-wheeled smartphone robot

Jiexin Wang[1] · **Eiji Uchibe**[2] · **Kenji Doya**[1,2]

Received: 7 July 2015 / Accepted: 16 December 2015 / Published online: 25 January 2016

Abstract This paper proposes a novel policy search algorithm called EM-based Policy Hyper Parameter Exploration (EPHE) which integrates two reinforcement learning algorithms: Policy Gradient with Parameter Exploration (PGPE) and EM-based Reward-Weighted Regression. Like PGPE, EPHE evaluates a deterministic policy in each episode with the policy parameters sampled from a prior distribution given by the policy hyper parameters (mean and variance). Based on EM-based Reward-Weighted Regression, the policy hyper parameters are updated by reward-weighted averaging so that gradient calculation and tuning of the learning rate are not required. The proposed method is tested in the benchmarks of pendulum swing-up task, cart-pole balancing task and simulation of standing and balancing of a two-wheeled smartphone robot. Experimental results show that EPHE can achieve efficient learning without learning rate tuning even for a task with discontinuities.

Keywords EM-based policy search · Reinforcement learning · Non-linear motor control · Smartphone robot

This work was presented in part at the 20th International Symposium on Artificial Life and Robotics, Beppu, Oita, January 21–23, 2015.

✉ Jiexin Wang
wang-j@oist.jp

[1] Kyoto University, Kyoto, Japan

[2] Okinawa Institute of Science and Technology, Okinawa, Japan

1 Introduction

Policy search methods which are widely adopted in high-dimensional robotic problems with continuous states and actions learn a policy directly from a reward function as an alternative to value function-based reinforcement learning [1]. Classic policy gradient algorithms such as REINFORCE [2] and GPOMDP [3] suffer from high variance in the gradient estimates because of the noise added at every time step for a stochastic policy.

Sehnke et al. proposed a method so-called Policy Gradients with Parameter-based Exploration (PGPE) [4] to solve this problem by evaluating deterministic policies with the parameters sampled from a prior distribution described by policy hyper parameters. The use of deterministic policy reduces the variance of the performance gradient with respect to the hyper parameters and the deterministic policy does not need to be differentiable. PGPE outperformed classical policy gradients algorithms, but still requires learning rate tuning for gradient ascent.

Peters and Schaal developed an EM-based Reinforcement Learning framework [5] to maximize the lower bound of the objective function. Depending on the choice of the probability distribution of the policy, the closed form solution for the maximization can be obtained.

Here, we propose a novel policy search algorithm called EM-based Policy Hyper Parameter Exploration (EPHE) that combines PGPE and an EM-based strategy to search for optimal hyper parameters of deterministic policies without the needs of gradient computation and learning rate tuning.

This method is designed in our smartphone robot project [6–8] for practical and efficient policy parameter learning. The project aims to develop a low-cost platform for multi-agent research. In the first stage, we developed a two-wheeled balancer as a single agent to evaluate various control and learning algorithm. In the real robot system, a deterministic policy is preferred over a stochastic policy to avoid unexpected behaviors and no need for differentiability allows a wider choice of control architectures with Parameter-based Exploration. Because learning without a huge number experience is also critical in hardware experiments, efficient update without learning rate tuning is also a highly favored feature of EM-based learning. We compare the EPHE mothed with PGPE [4] and Finite Difference method [3] in benchmark tasks of pendulum swing-up with limited torque, cart-pole balancing, and our two-wheeled smartphone robot simulator. Results show that EPHE outperforms the previous policy search methods.

2 Learning method

2.1 REINFORCE, PGPE and EM-based algorithms

We assume a standard discrete-time Markov Decision Process (MDP) setting. At each time step t, an agent takes an action u_t based on a state x_t according to the policy $\pi(u_t|x_t,\theta)$ parameterized by a vector θ. The environment makes a transition to a next state x_{t+1} according to $p(x_{t+1}|x_t,u_t)$ and gives a scalar reward r_t to the agent. We denote a state-action-reward sequence as $h=[x_1,u_1,r_1,\ldots,x_T,u_T,r_T,x_{T+1}]$. The goal of reinforcement learning is to find the parameter θ that maximizes an objective function defined as the agent's expected reward

$$J(\theta)=\int p(h|\theta)R(h)\mathrm{d}h \tag{1}$$

where $R(h)$ is the cumulative reward of the sequence h, and $p(h|\theta)$ is the probability to observe h. Under the Markovian environmental assumption, $p(h|\theta)$ is given by:

$$p(h|\theta)=p(x_1)\prod_{t=1}^{T}p(x_{t+1}|x_t,u_t)\pi(u_t|x_t,\theta). \tag{2}$$

To maximize $J(\theta)$, one way is to estimate the gradient $\nabla J(\theta)$ to perform gradient ascent. REINFORCE [2] obtains the gradient by estimating $\nabla_\theta \log p(h|\theta)$ directly, which yields

$$\nabla_\theta J(\theta)=\int_H p(h|\theta)\nabla_\theta \log p(h|\theta)R(h)\mathrm{d}h. \tag{3}$$

Substituting (3) with (2), we have

$$\nabla_\theta J(\theta)=\int_H p(h|\theta)\sum_{t=1}^{T}\nabla_\theta \log\pi(u_t|x_t,\theta)R(h)\mathrm{d}h.$$

Although it is not practical to integrate over the entire space of histories, we can use sampling to obtain the estimate

$$\nabla_\theta J(\theta)\approx\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log\pi(u_t^n|x_t^n,\theta)R(h^n)$$

where N denotes the number of histories. To reduce the variance of the gradient, $R(h)$ can be replaced with $R(h)-b$, where b is a reward baseline [5].

The problems of REINFORCE are that the policy has to be differentiable with respect to the policy parameters and that evaluation of a stochastic policy can lead to a high variance in the entire histories.

PGPE [4] addresses these problems by considering a distribution of deterministic policies with the policy parameters θ sampled from a prior distribution defined by the hyper parameters ρ, which are typically the mean and the variance of θ. The objective function is given by

$$J(\rho)=\int_\Theta\int_H p(h|\theta)p(\theta|\rho)R(h)\mathrm{d}h\mathrm{d}\theta. \tag{4}$$

It should be noted that the variance of $p(h|\theta)$ of PGPE can be kept small because a deterministic policy is adopted. By updating the hyper parameter vector ρ, we can obtain the deterministic policy $\pi(u_t|x_t,\theta)$ where θ is eventually computed by the expectation of the prior distribution.

Differentiating Eq. (4) with respect to ρ gives us

$$\nabla_\rho J(\rho)=\int_\Theta\int_H p(h|\theta)p(\theta|\rho)\nabla_\rho \log\rho(\theta|\rho)R(h)\mathrm{d}h\mathrm{d}\theta.$$

Again by sampling, we have the gradient estimate

$$\nabla_\rho J(\rho)\approx\frac{1}{N}\sum_{n=1}^{N}\nabla_\rho \log p(\theta|\rho)R(h^n)$$

For PGPE to perform gradient ascent, a learning rate parameter has to be tuned.

EM-based Policy Search [9] estimates a lower bound of the expected return from histories and iteratively updates the policy parameter using an analytic solution for the maximum of the lower bound. This way, there is no learning rate parameter for an EM-based method. The detail of the EM-based method is illustrated below in the context of our hyper parameter learning.

2.2 Proposed method

Here we describe our proposed method, EM-based Policy Hyper Parameter Exploration (EPHE) by integrating the features of PGPE [4] and EM-based Policy Search [9]. To establish the lower bound, we consider a new parameter distribution over hyper parameter vector ρ'. Using Jensen's inequality under the assumption that $R(h)$ is strictly positive, we have the log ratio of two objective functions

$$\log\frac{J(\rho')}{J(\rho)} = \log\int_{\Theta}\int_{H}\frac{R(h)p(h|\theta)p(\theta|\rho)}{J(\rho)}\frac{p(\theta|\rho')}{p(\theta|\rho)}\mathrm{d}h\mathrm{d}\theta$$
$$\geq \int_{\Theta}\int_{H}\frac{R(h)p(h|\theta)p(\theta|\rho)}{J(\rho)}\log\frac{p(\theta|\rho')}{p(\theta|\rho)}\mathrm{d}h\mathrm{d}\theta$$

Hence, the lower bound is defined by

$$\log J_L(\rho') = \log J(\rho) + \int_{\Theta}\int_{H}\frac{R(h)p(h|\theta)p(\theta|\rho)}{J(\rho)}\log\frac{p(\theta|\rho')}{p(\theta|\rho)}\mathrm{d}h\mathrm{d}\theta^{\cdot} \quad (5)$$

To maximize this lower bound, the derivative of (5) with respect to ρ' should equal to zero

$$\nabla_{\rho'}\log J_L(\rho') = \int_{\Theta}\int_{H}\frac{R(h)p(h|\theta)p(\theta|\rho)}{J(\rho)}\nabla_{\rho'}\log\frac{p(\theta|\rho')}{p(\theta|\rho)}\mathrm{d}h\mathrm{d}\theta = 0.$$

Since $J(\rho)$ is constant, this equation can be simplified as

$$\int_{\Theta}\int_{H} R(h)p(h|\theta)p(\theta|\rho)\nabla_{\rho'}\log p\left(\theta|\rho'\right)\mathrm{d}h\mathrm{d}\theta = 0.$$

By applying sampling trick again, we have

$$\frac{1}{N}\sum_{n=1}^{N}\nabla_{\rho'}\log p\left(\theta^n|\rho'\right)R\left(h^n\right) = 0. \quad (6)$$

If $p(\theta|\rho')$ is represented by an exponential family distribution, the update rule is given by a closed form. In particular, we consider that $p(\theta|\rho')$ is given by a product of independent Gaussian distributions $N(\theta_i|\eta_i', {\sigma_i'}^2)$ for each parameter θ_i in θ. The log derivatives of $p(\theta|\rho')$ with respect to η_i' and σ_i' are computed as

$$\nabla_{\eta_i'}\log p\left(\theta|\rho'\right) = \frac{\theta_i - \eta_i'}{{\sigma_i'}^2} \quad (7)$$

$$\nabla_{\sigma_i'}\log p\left(\theta|\rho'\right) = \frac{(\theta_i - \eta_i')^2 - {\sigma_i'}^2}{{\sigma_i'}^3} \quad (8)$$

Substituting Eqs. (7) and (8) into Eq. (6) yields

$$\eta' = \frac{\sum_{n=1}^{N}[R(h^n)\theta_i^n]}{\sum_{n=1}^{N}R(h^n)}$$

$$\sigma' = \sqrt{\frac{\sum_{n=1}^{N}\left[R(h^n)\left(\theta_i^n - \eta_i'\right)^2\right]}{\sum_{n=1}^{N}R(h^n)}} \quad (9)$$

It should be noted that the denominator is positive because we assume that $R(h)$ is strictly positive so that it can resemble an (improper) probability distribution to weight the parameters. To obtain a good sampling performance, we only take the parameters from the best K returns in N trajectories for updating.

```
**Algorithm** EM-based Policy Hyper Parameter Exploration
**Input**: initialize policy hyper parameters $\eta$ and $\sigma$
**Repeat**:
  Perform $N$ trajectories:
        for each trajectory
        draw $\theta_i^n \sim N(\eta_i, I\sigma_i^2)$ for all $i$
        evaluate $R(h^n)$
  Select $K$ best trajectories from the sorted $R(h^n)$
  Update
        $\eta = \frac{\sum_{k=1}^{K}[R(h^k)\theta_i^k]}{\sum_{k=1}^{K}R(h^k)}$
        $\sigma = \sqrt{\frac{\sum_{k=1}^{K}[R(h^k)(\theta_i^k - \eta_i)^2]}{\sum_{k=1}^{K}R(h^k)}}$
**Until** Convergence
```

3 Experiments

In this section, we compare our method EPHE with PGPE [4] and classic policy gradient method Finite Difference (FD) [3]. For each method we use $N = 20$ trajectories to update one set of parameters, and select $K = 10$ to obtain the elite parameters for updating in our method. The results are taken by the average of 20 independent runs. We plot the learning curves of the average and the standard error of cumulative returns against the iterations of parameter updating.

3.1 Pendulum swing-up with limited torque

The target of this non-linear control task is to swing up the pendulum to the upright position and stay as long as possible [10]. We use 16*16 radial basis functions to represent the two-dimensional state variables, the angle, and the angular velocity of the pendulum: $\boldsymbol{x} = \{\varphi, \dot{\varphi}\}$. The action is the torque applied to the pendulum $u = 5 * \tanh\left(\boldsymbol{\theta}^T\Phi(\boldsymbol{x})\right)$ with maximum torque 5 [N*m], where $\boldsymbol{\theta}$ is the policy parameter and $\Phi(\boldsymbol{x})$ is the basis function vector. The system

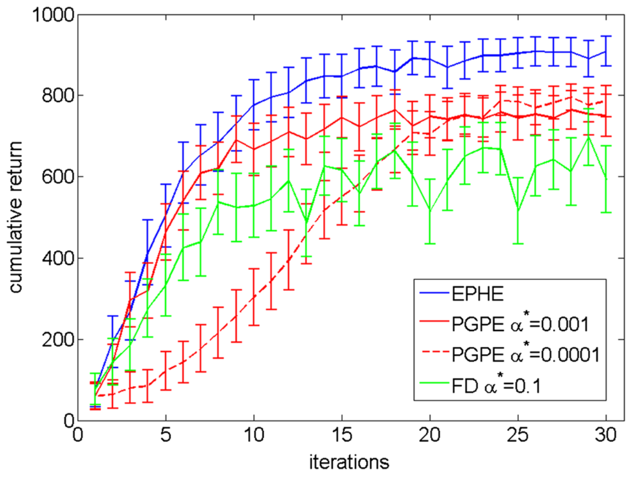

Fig. 1 Learning curves of the swing-up task

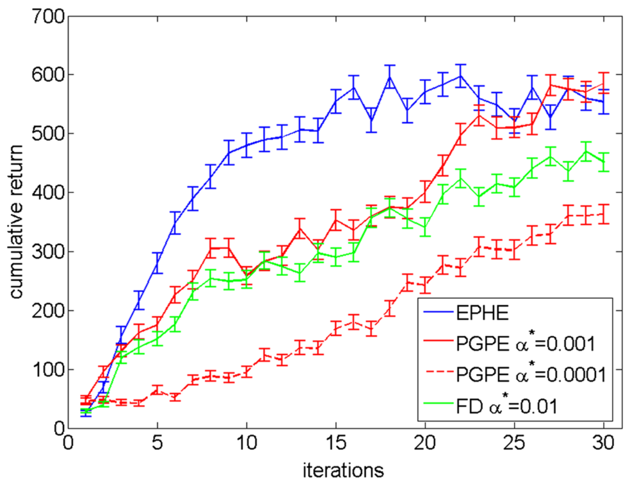

Fig. 2 Learning curves of the cart-pole task

starts from an initial state $\boldsymbol{x}_0 = \{\varphi_0, 0\}$, where φ_0 is randomly selected from $[-\pi, \pi]$ [rad], and terminates when $|\dot{\varphi}| \geq 4\pi$ [rad/s]. The sampling rate is 0.02 [s] for each time step and maximum time steps is 1000 (=20 [s]) for one episode. The strictly positive reward for one history is given by

$$R(h) = \sum_{t=1}^{T} \exp(-x^T Q x - u^T R u) \tag{10}$$

where Q and R are the quadratic penalty matrix determined by users. For Finite Difference, the initialization of policy parameters is $\boldsymbol{\theta}_0 = 0$, and the steps for the policy parameter update are $\delta\boldsymbol{\theta} \sim U(-3.46, 3.46)$, a uniform distribution with variance 1. For PGPE and our method EPHE, the initial hyper parameters are $\boldsymbol{\eta}_0 = 0, \boldsymbol{\sigma}_0 = 1$.

Figure 1 shows the performance of FD, PGPE and our method. We hand-tuned the learning rates for each method and found that separate learning rates for each parameter are required for PGPE. The optimal parameters were $\alpha = 0.1$ for FD, and $\alpha_\eta = 0.001$, $\alpha_\sigma = 0.0001$ for PGPE. We also showed the performance of PGPE with $\alpha_\eta = 0.0001$, $\alpha_\sigma = 0.0001$ to illustrate its parameter dependence. The proposed method learned faster and achieved better performance after 30 iterations without the need for tuning the learning rate.

3.2 Cart-pole balancing

In this task, the agent aims to maximize the length of time of a movable cart balancing a pole upright in the center of a track [11]. The state variables are the position and the velocity of the cart on the track, and the angle and the angular velocity of the pole: $\boldsymbol{x} = \left\{x, \dot{x}, \theta, \dot{\theta}\right\}$. The action is the force applied to the cart given by a linear parameterized policy $u = \boldsymbol{\theta}^T \boldsymbol{x}$. We add Gaussian white noise with standard deviation of 0.001 [rad/s] and 0.01 [m/s] to the dynamics. The system starts within a random position and a random angle inside $[-0.2, +0.2]$ [rad], and $[-0.5, +0.5]$ [m] until it reaches the target region of $[-0.05, +0.05]$ [rad] and $[-0.05, +0.05]$ [m], and terminates at $|x| \geq 2.4$ [m], and $|\theta| \geq 0.7$ [rad]. The sampling rate is 0.02 [s] for each time step and maximum time steps is 1000 (=20 [s]) for one episode. The strictly positive reward is the same as (10). The initializations of policy parameters for FD and hyper parameters for PGPE and EPHE are from a reasonable prior knowledge, which indicates certain distance from the optima. The parameter update for the FD controller is $\delta\boldsymbol{\theta} \sim U(-3.9, 3.9)$, a uniform distribution with variance 5. We tested with the same initialization as FD of $\boldsymbol{\eta}_0$ with different $\boldsymbol{\sigma}_0 = 5$ for PGPE, and $\boldsymbol{\sigma}_0 = 35$ for EPHE.

Figure 2 shows the performance of FD, PGPE and our method. The best learning rates were, $\alpha = 0.01$ for FD, $\alpha_\eta = 0.001$, $\alpha_\sigma = 0.0001$ for PGPE. Our method EPHE achieved faster learning without learning rate tuning.

3.3 Two-wheeled smartphone robot

The goal of the smartphone robot project is to construct an affordable, high-performance multi-agent platform for researching on robot social behaviors [6]. Even as a single agent, it has lots of possibilities to achieve various behaviors for testing and developing motor control algorithms under control theory and Reinforcement Learning domain. In our previous work [7, 8], we developed a two-wheel balancer and successfully realized standing-up and balancing behavior by a switching control architecture with an optimal linear controller and a hand-tuned non-linear controller. With our new method EPHE, the robot is expected to optimize the policy parameters automatically in a more practical and efficient fashion. In this section, we compared our method with PGPE and FD in our two-wheeled smartphone robot simulator.

The state variables are tilting angle and angular velocity of the body, and rotating angle and angular velocity of the wheel where $\boldsymbol{x} = \{\varphi, \dot{\varphi}, \vartheta, \dot{\vartheta}\}$. The control input u is the motor torque applied to the left and right wheel. We adopt a switching framework in which a linear feedback stabilizer is selected to achieve balancing if the tilting angle of the robot body is within the range of $[-\varphi_s, \varphi_s]$, otherwise the CPG-based destabilizer is applied. The policy parameters are the four-dimensional control gain vector for the linear stabilizer, the switching threshold, and two parameters of the oscillator: $\boldsymbol{\theta} = \{k_1, k_2, k_3, k_4, \varphi_s, \omega, \beta\}$. Figure 3 shows the architecture. We also added observation Gaussian white noise with standard deviation of 0.01 to the system.

The agent is required to start moving from the resting angle 60°, bounce with the bumper to stand up and finally achieve balancing. The simulation runs with a sampling rate 0.02 [s] for each step. The agent learns one episode within the maximum of 1000 steps (=20 [s]). The cumulative reward is the same as (10). We initialize the parameters $\boldsymbol{\theta}_0$ and the step size $\delta\boldsymbol{\theta}$ for FD, and the hyper parameters $\boldsymbol{\eta}_0$ for PGPE and our method with uniform distributions, and fixed $\boldsymbol{\sigma}_0$ based on the prior knowledge we obtained in [7].

Figure 4 shows the learning performance. The best learning rates are, $\alpha_k = 0.0001$, $\alpha_{\varphi_s} = 0.00001$, $\alpha_\omega = 0.01$, $\alpha_\beta = 0.01$ for FD, $\alpha_\eta = 0.001$, $\alpha_\sigma = 0.001$ for PGPE. The success rates of each method are illustrated in Table 1. Our method outperformed others after 10 iterations and achieved a more reliable performance after 20 iterations.

We also plot the distribution of 20 final optimized parameters in Fig. 5. FD has the most centralized distribution of final optimized parameters because it represents the policy parameters while PGPE and EPHE represent the distribution of the policy parameters. We pick up one set of the optimized parameters $\boldsymbol{\eta} = \{0.0021, 0.0982, 0.2953, 0.0651, 47^o, 11.7169, 18.7890\}$ based on our method and sample 10 sets of policy parameters based on the Gaussian prior distribution to illustrate the bouncing and stabilizing behaviors in Fig. 6. It shows that the control signals are synchronized with the angular velocity and the switcher can successfully coordinate the two controllers to achieve the expected behaviors.

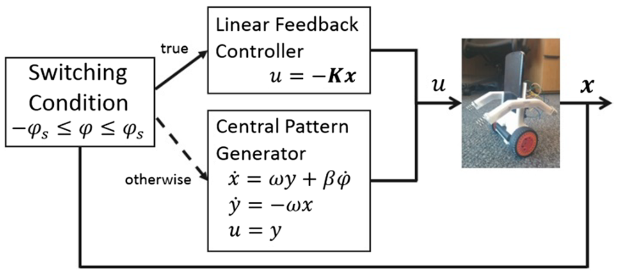
Switching Condition
$-\varphi_s \le \varphi \le \varphi_s$
true
otherwise
Linear Feedback Controller
$u = -\boldsymbol{K}\boldsymbol{x}$
Central Pattern Generator
$\dot{x} = \omega y + \beta\dot{\varphi}$
$\dot{y} = -\omega x$
$u = y$
u
$\boldsymbol{x}$

Fig. 3 Switching control architecture of smartphone robot

Table 1 Successful rate of each method

EPHE	90 %
PGPE	60 %
Finite difference	85 %

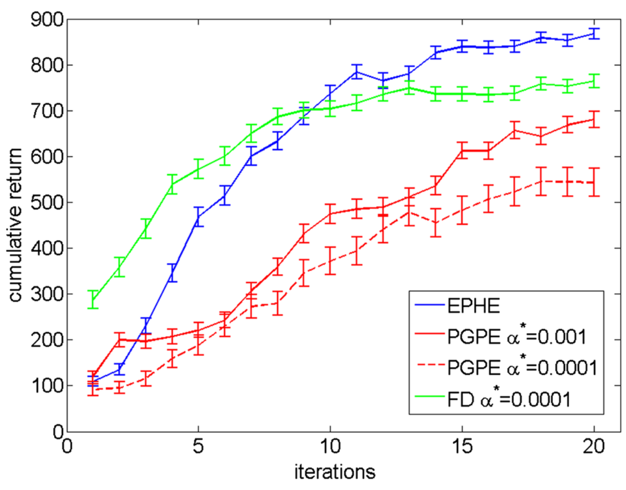

Fig. 4 Learning curves of the smartphone robot simulator

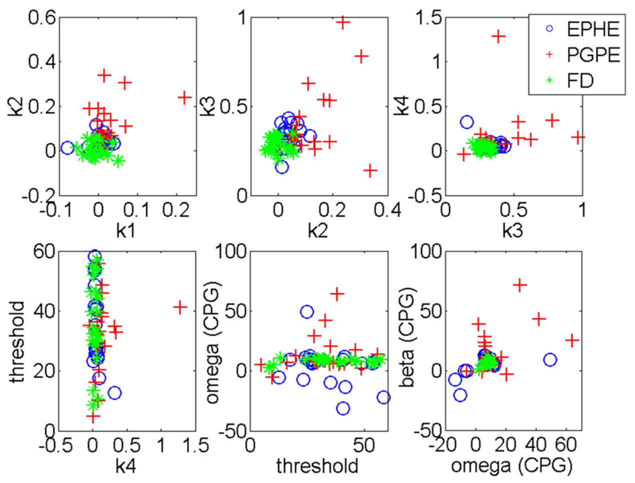

Fig. 5 Distributions of the optimized parameters in the case of the smartphone robot experiment

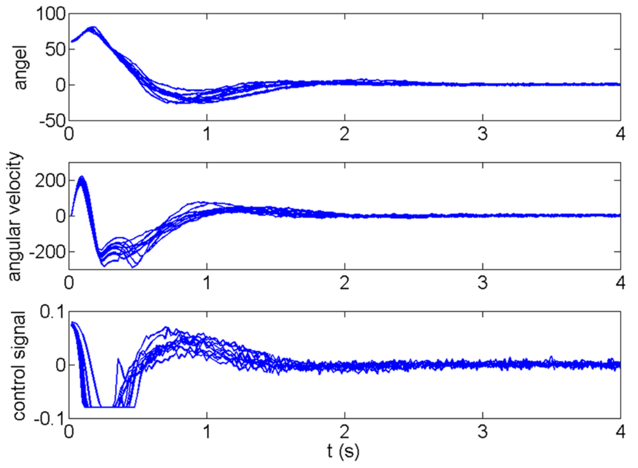

Fig. 6 Trajectories realized by the policy of which the parameters are sampled from the optimized prior distribution

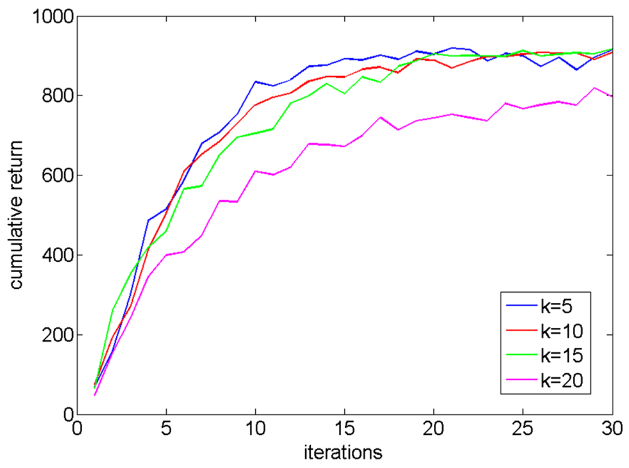

Fig. 7 Sensitivity of K to N in the swing-up task

4 Discussion

Our method EPHE can compete the other two methods in all the tasks with a faster convergence speed and a steadily higher return (Figs. 1, 2 and 4). In the smartphone robot simulator case, Finite Difference method learns better in the beginning because it searches in the policy parameters space directly under one-dimensional uniform distribution domain, and it is easy to be trapped in local optima. This is illustrated in Fig. 5.

We also found that the difference between the initialization of the variances illustrates different insights of PGPE and our method: PGPE optimizes the hyper parameters by computing gradients, in which case, smaller variance leads to a more precise approximation. While our method computes the average of the sampled points which suggests larger initial variance explores more. The learning behavior is much improved by the K-elite selection mechanism. Because parameters are quite far away from the optima in the beginning of learning, frequent failures will slow down the learning process. Also, it is easy to be trapped in local optima by updating with all the parameters (when $K = N$). Figures 7, 8 and 9 reveal the sensitivity of K to N in three tasks. Agents with smaller setting of K learn relatively faster but reach no better performance in the end. There is no significant difference between different settings of K, but agents without the selection mechanism achieve much worse behaviors. Another interesting finding is, in the Android robot case, the threshold of switching condition seems not crucial parameters to be tuned. This saves the burden of making different arms for the robot body in real hardware.

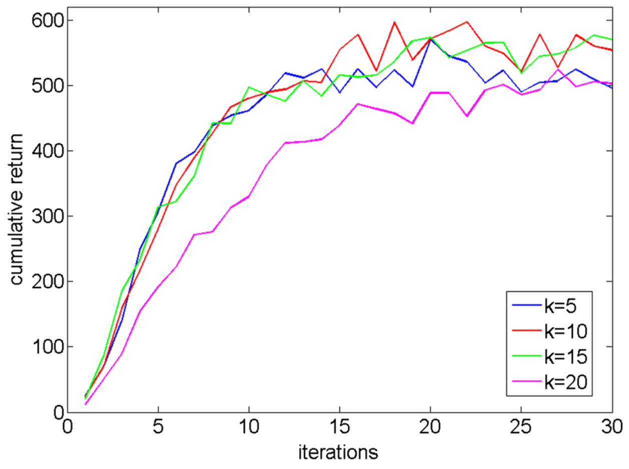

Fig. 8 Sensitivity of K to N in the cart-pole task

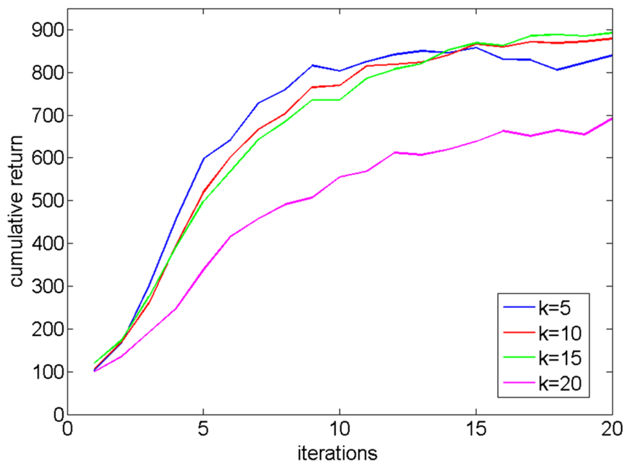

Fig. 9 Sensitivity of K to N in the smartphone robot task

5 Conclusion and future work

In this paper, we developed a new policy search algorithm, EM-based Policy Hyper Parameter Exploration (EPHE). We tested it in two benchmark tasks and our two-wheeled Android phone robot simulator with a non-linear, non-differentiable controller. Our method integrated PGPE with the EM-based update that maximizes a lower bound of the expected return in each iteration of hyper parameter updating. Simulation results showed that our method outperforms other policy gradient methods such as Finite Difference and PGPE after fine tuning of the learning rates. The advantages of our method are: (1) the controller is deterministic, (2) it does not require the controller to be differentiable, (3) it avoids hand-tuning of the learning rate, which are highly favored in practical robot systems.

For future work, we will improve our method by taking into account the correlation of the parameters and developing more sophisticated sampling methods. We notice that our algorithm with a Gaussian prior is similar to the CMA-ES optimization method [12], and further comparisons with recent policy search methods such as the path integral framework [13] are also required. We will test our method in the real robot system to realize efficient tuning-free performance.

Acknowledgments This work was supported by MEXT/JSPS KAKENHI Grant number 24500249 and 26120727.

References

1. Deisenroth MP, Neumann G, Peters J (2013) A survey on policy search for robotics. Found Trends Robot 2(1–2):1–142
2. Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8:229–256
3. Peters J, Schaal S (2006) Policy gradient methods for robotics. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems. Beijing, China
4. Sehnke F, Osendorfer C, Rückstieß T, Graves A, Peters J, Schmidhuber J (2010) Parameter-exploring policy gradients. Neural Netw 21(4):551–559
5. Peters J, Schaal S (2007) Reinforcement learning by reward-weighted regression for operational space control. In: IEEE Proceedings of the IEEE International Conference on Intelligent Robots and Systems. Piscataway, NJ
6. Yoshida N, Yoshimoto J, Uchibe E, Doya K (2012) Development of robot platform with smart phone. In: Proceedings of the annual conference on Robotics Society of Japan. (in Japanese)
7. Wang J, Uchibe E, Doya K (2013) Standing-up and balancing behaviors of android phone robot, Technical committee on Nonlinear Problems, IEICE, Hong Kong, China
8. Wang J, Uchibe E, Doya K (2014) Control of two-wheel balancing and standing-up behaviors by an android phone robot. In: Proceedings of the annual conference on Robotics Society of Japan
9. Hachiya H, Peters J, Sugiyama M (2011) Reward-weighted regression with sample reuse for direct policy search in reinforcement learning. Neural Comput 23:2798–2832
10. Doya K (2000) Reinforcement learning in continuous time and space. Neural Comput 12(1):219–245
11. Riedmiller M, Peters J, Schaal S (2007) Evaluation of Policy Gradient Methods and Variants on the Cart-Pole Benchmark. In: Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning
12. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evolut Comput 9(2):159–195
13. Theodorou E, Buchli J, Schaal S (2010) A generalized path integral control approach to reinforcement learning. J Mach Learn Res 11:3137–3181