Paper:

An Efficient Adaptation Method for Model-Based Meta-Reinforcement Learning

Shiro Usukura, and Hajime Nobuhara

University of Tsukuba, 1-1-1 Tennoudai, Tsukuba, 305-8573 Ibaraki, Japan
E-mail: usukura@cmu.iit.tsukuba.ac.jp

[Received ; accepted]

We propose an efficient adaptation method for model-based meta-reinforcement learning by combining two conventional methods. Model-based meta-reinforcement learning can achieve high performance by adapting observed data in every time step in unstable environments. There are two conventional adaptation methods for model-based meta-reinforcement learning which are KShot-adaptation method and the Continued-adaptation method. These methods have similar algorithms, but different characteristics. KShot-adaptation method tends to underfit observed data, while Continued-adaptation method tends to overfit it. Therefore, we propose a method that combines these two conventional methods by introducing a new parameter. We conducted evaluation experiments in a half-cheater-disable-joint environment that reproduces changes of dynamics by disabling the joints. As a result, our method obtained an average reward of 3.9% higher than the KShot-Adaptation method and 31% higher than the Continued-Adaptation method.

Keywords: Meta reinforcement learning, Online learning, Deep Learning

1. Introduction

In recent years, applications of reinforcement learning of robot control field have been actively studied, and complicated control has become possible. However, most of them are the result under the fixed environment in simulator or laboratory, and applications in the real world are seldom carried out at present. In the real world, following two points are problems: 1) it is difficult to collect a large amount of data, and 2) agents and environments can change. Factor 1. the difficulty of collecting large amounts of data can be addressed by using model-based methods. Reinforcement learning can be classified into two types, model-based and model-free. In general, it is said that model-based methods have higher sampling efficiency but lower performance than model-free methods. However,Some studies have shown that the performance can be improved to the same or higher level as model-free by using ensemble methods [3, 7]. Model-based reinforcement learning learns a dynamics model and selects actions by model predictive control (MPC). As a representative MPC, there are random shooting (RS) and cross entropy method (CEM) [2].

Next, we describe factor 2. changes of agent and environments. Reinforcement learning learns the optimal policy by repeating trials. Therefore, high performance can be expected in a fixed environment, but it can not be expected in an unstable environment. In the real world, changes of environment and agent can occur frequently. Taking a walking robot as an example, the failure of the manipulator corresponds to an agent change, and a change of materials and condition of the floor surface corresponds to an environment change. In model-based reinforcement learning, it is possible to deal with these changes by re-learning the dynamics model from scratch but re-learning requires an enormous amount of time. Therefore, Nagabandi et al. proposed a method to enable efficient adaptation by using MAML [4] which is a meta-learning method (We call this adaptation method KShot-Adaptation method.). In unstable environments, this method achieved high performance in comparison with simple model-based reinforcement learning. But there was also a problem that adaptation information was not propagated to the next time step. In order to solve this problem, a simple extended method by online learning is proposed in a subsequent paper [6] (We call this adaptation method Continued-Adaptation method.). Because of the iterative updating, this method has a problem of overfitting to currently observed data. And, it has been suggested that whether the KShot-Adaptation method or Continued-Adaptation method is superior depends on the problem setting [6].

In this paper, we propose an adaptation method between the KShot-Adaptation method and the Continued-Adaptation method. This method is designed with the expectation that the problems of the two conventional methods would suppress each other. We experimentally show that the proposed method earns on average 3.9% higher rewards than KShot-Adaptation method and 31% higher rewards than Continued-Adaptation method.

2. Rerated Work

Reinforcement learning is susceptible to changes of environmental and agent. Therefore, the research which

adapts observed data in every time step using meta reinforcement learning is carried out [1, 5]. However, these methods may not be able to perform efficient adaptation because adaptation information is not propagated to the next time. In a subsequent paper, Two methods have been proposed to deal with this problem [6]. The first is a simple extended method by online learning, and the second is a method which automatically adjust step size according to the importance of observation data. The second is called MOLe.

3. Proposed Method

We propose an efficient model adaptation method for model-based meta-reinforcement learning by combining KShot-Adaptation method and Continued-Adaptation method.

Let $\tau(i,j), i < j$ be states from time i to time j+1 and actions from time i to time j. That is, $\tau(i,j)$ is defined as follows:

$$\tau(i,j) \equiv \{s_i, a_i, s_{i+1}, a_{i+1}, s_j, a_j, s_{j+1}\}, \quad \ldots \quad (1)$$

The dynamics model $\hat{p}_\theta(s_{i+1}, |s_i, a_i)$ is a model which predicts the next state from current state and action. And, θ is a model parameter. The Loss $\mathscr{L}$ is difined as:

$$\mathscr{L}(\tau(i,j),\theta) \equiv -\frac{1}{j-i}\sum_{\tau=i}^{j} \hat{p}_\theta(s_{\tau+1}, |s_\tau, a_\tau), \quad . \quad (2)$$

θ'_t is a model parameter which adapted to observed data at time t. θ'_t is calculated by KShot-Adaptation method as follows:

$$\theta'_t = \theta^* - \alpha\nabla_\theta \mathscr{L}(\tau(t-M,t-1),\theta^*), \quad \ldots \quad (3)$$

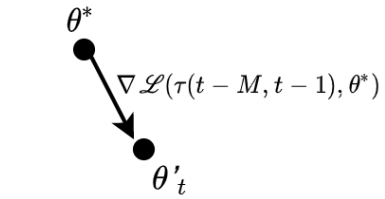
where θ^* is a optimal model parameter obtained by learning, and α is learning rate [5]. In eq.3, a model parameter θ'_{t-1} which adapted in previous time is not used. An efficient adaptation may not be carried out, because θ'_{t-1} is likely to be beneficial.

On the other hand, Continued-Adaptation method updates the parameter continually. That is, θ'_t is calculated by Continued-Adaptation method as follows:

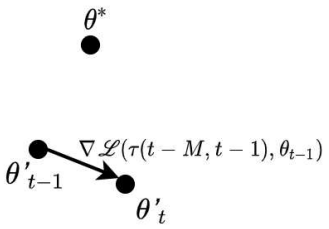
$$\theta'_t = \theta'_{t-1} - \alpha\nabla_\theta \mathscr{L}(\tau(t-M,t-1),\theta_{t-1}), \quad \ldots \quad (4)$$

where α is learning rate [6]. Unlike the KShot-Adaptation method, Continued-Adaptation method uses previous adapted parameter θ_{t-1}. Therefore, an efficient adaptation can be expected. However, it may over-adapt to observed data. The system may not be able to respond if an agent or environment suddenly changes. And, it has been suggested that whether the KShot-Adaptation method or Continued-Adaptation method is superior depends on problem settings [6].

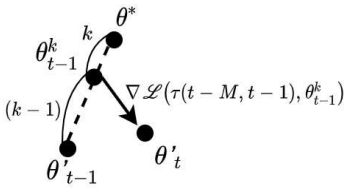
In this paper, we introduce new parameter $k \in [0,1]$ to combine KShot-Adaptation method and Continued-Adaptation method. This method calculates θ_{t-1} based on θ^k_{t-1} which is obtained by internally dividing line segment $\theta^*\theta'_{t-1}$ by k:(1-k). Specifically, updating formulas



(a) KShot-Adaptation method



(b) Continued-Adaptation method



(c) Proposed method

Fig. 1. : Updating model Parameter for adaptation

is as follows:

$$\theta^k_{t-1} = (1-k)\theta^* + k\theta'_{t-1}, \quad \ldots \quad (5)$$

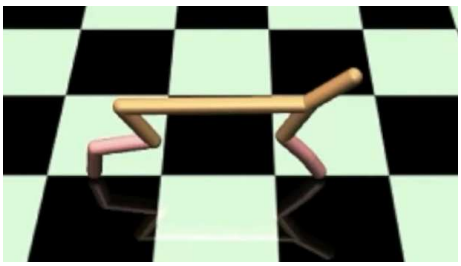
$$\theta'_t = \theta^k_{t-1} - \alpha\nabla_\theta \mathscr{L}(\tau(t-M,t-1),\theta^k_{t-1}), \quad \ldots \quad (6)$$

where α is learning rate. Unlike the two conventional methods, the proposed method uses both θ^* obtained by learning and θ'_{t-1} adapted to previous time step. And, the ratio of both can be adjusted by parameter k. That is, the propagation rate of previous adaptive information can be adjusted by parameter k. In particular, it corresponds to KShot-Adaptation method for $k = 0$ and Continued-Adaptation method for $k = 1$. The proposed method is expected to suppress the problems of the two conventional methods. But optimal value of k is considered to vary depending on problem settings.

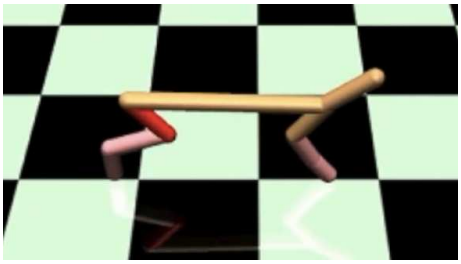
4. Experiment

4.1. Simulation environment

We conducted evaluation experiments using the HalfCheetah: disabled joint environment [5] as shown in **Fig.2**. The state is $s \in \mathbb{R}^{20}$and the action is $a \in \mathbb{R}^6$. elements of the action a are values of the torque for each joint. Each joint is numbered as shown in **Fig.3**, and can be disabled. To disable a joint causes a change of dynamics. And this change is expected to be different depending



(a) Every joint is enabled



(b) #1 joint is disabled

Fig. 2. : HalfCheetah: disabled joint

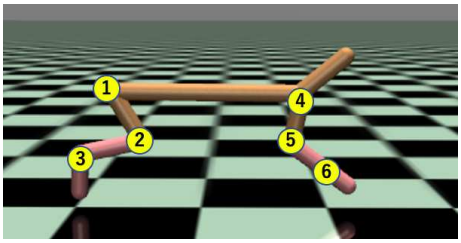


Fig. 3. : Joint number

on the joint to be disable.

When We train dynamics model, one episode is 500 steps, and all joints were allowed to move at the beginning of episodes as shown in **Fig.2a**. At each time step, randomly selected joint with a probability of $c = 0.005$ was disabled. However, when the already disabled joint exists, it is kept as it is, and the new joint is not disabled. That is, at time t, the probability that a disabled joint exists is $p_t = 1-(1-c)^t$.

When we evaluate the proposed adaptation method, one episode is 800 steps, and all joints were allowed to move at the beginning of the episode. Then, at step 400, #1 joint was disabled as shown in **Fig.2b**.

4.2. Implementation

A trajectory $\tau(t-M, t+K)$ is set to $M = 16$, $K = 16$. And, a dynamics model is constructed by a multilayer perceptron network (MLP) with 2 hidden layers, and ReLU function is used for the activation function. Using CEM for MPC, a planning horizon H is set to 10, and sampling number N is set to 512.

Table 1. : Mean and standard deviation of rewards (50 episodes)

(a) In the first 400 steps

	k					
	0.0	0.0625	0.125	0.25	0.5	1.0
Mean	297	309	299	304	308	287
SD	6.10	5.95	5.63	5.59	5.85	5.19

(b) In the last 400 step

	k					
	0.0	0.0625	0.125	0.25	0.5	1.0
Mean	271	281	280	248	199	164
SD	5.48	5.34	5.40	5.81	5.49	6.82

(c) In the while episode

	k					
	0.0	0.0625	0.125	0.25	0.5	1.0
Mean	568	590	579	552	507	450
SD	7.02	6.37	6.40	6.20	5.22	6.84

4.3. Result

In evaluation environment setting described in Chapter 4.1, we ran 50 episodes for each parameter $k \in \{0, 0.625, 0.125, 0.25, 0.5, 1\}$. We compared the mean and standard deviation of rewards earned in one episode, as well as the mean and standard deviation for the first 400 steps before disabling joint, and for the second 400 steps after joint disabling. From **Table 1a**, rewards in the first 400 steps before the dynamics change occurred hardly changed by the parameter k. On the other hand, from **Table 1b** rewards in the second 400 steps after the dynamics change changed depending on the value of the parameter k, and high rewards were obtained at $k = 0.0625,\ 0.125$. In addition, average rewards earned over an episode become the maximum when parameter $k = 0.0625$. At this parameter, the rewards was 3.9% higher than KShot-Adaptation method ($k = 0$), and 31% higher than Continued-Adaptation method ($k = 1$). And, **Table 1** indicated that the performance changed continuously according to k.

5. Conclusion

We propose an efficient model adaptation method in model-based meta reinforcement learning using MAML. The proposed method is combined KShot-Adaptation method and Continued-Adaptation method. As a result of the experiment in the unstable environment, the proposed method obtained 3.9% higher rewards than KShot-Adaptation method ($k = 0$), and 31% higher rewards than Continued-Adaptation method ($k = 1$) at an appropriate value of k.

In this paper, k is given as a hyperparameter. However,

it is also assumed that optimal k can change during an episode. Therefore, We will examine a technique which adjusts *k* during an episode like MOLe [6].

References:

[1] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine. "Model-Based Meta-Reinforcement Learning for Flight with Suspended Payloads", 2020.

[2] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and F. O. I. Engineering. "The Cross-Entropy Method for Optimization".

[3] K. Chua, R. Calandra, R. McAllister, and S. Levine. "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models", 2018.

[4] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks", CoRR, abs/1703.03400, 2017.

[5] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. "Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning", 2019.

[6] A. Nagabandi, C. Finn, and S. Levine, "Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL", CoRR, abs/1812.07671, 2018.

[7] T. Wang and J. Ba. "Exploring Model-based Planning with Policy Networks", 2019.