Reptile: First-Order Meta-Learning Implementation for Pendulum Reinforcement Learning Problem

Nguyen, Q., Ngo, V., & Chung, T. (2018). *Reptile: First-Order Meta-Learning Implementation for Pendulum Reinforcement Learning Problem*.

Reptile – First-Order Meta-Learning Implementation for Pendulum Reinforcement Learning Problem

Quang Nguyen[1], Vien Ngo[2], TaeChoong Chung[3]
[1,3] Department of Computer Science and Engineering, Kyung Hee University, South Korea
[2] School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK
quangnd@khu.ac.kr, v.ngo@qub.ac.uk, tcchung@khu.ac.kr

Abstract

Meta Learning has been grabbed much attentions recently after a lot of significant improvements in deep learning. As usual, the deep learning neural network is trained from scratch and learn to handle a specific task according to knowledge gained from very large number of observations presented to the system during the training time. Although the trained model can handle the trained task properly after sufficient training, it hardly keeps its performance to the few-shot learning problem. Meta learning is introduced as an learning method for the network so that it can be trained in condition of data sparsity for faster convergence compared to learning from scratch. Introduced in 2018, Reptile, a simple algorithm for First-Order Meta-Learning, has shown its optimality as an initialization method in dealing with an entire task distribution. To address the challenge of applying Reptile to Reinforcement Learning problems as stated in its original paper, the authors will propose an implementation of Reptile to Pendulum environment with the hope to propagate this algorithm in Reinforcement Learning area.

Keywords: Meta-Learning, Neural Network Initialization, First-Order Meta-Learning, Reptile, Reinforcement Learning

1. INTRODUCTION

Learning from perspective of machines is a process to generalize and obtain knowledge from environment's observations. Normally, a neural network with some specific architecture will be trained with a large volume of data to fit to the distribution of a given task. The outcome after this training is a model that can mimic the characteristics or react with high accuracy to the given distribution.

However, as human can learn new skills very quickly after a short period of training, machines are also placed under the expectation that its learning process could approach similarly to the learning level of human. This demand requires efforts in solving two main problems. The first is the ability to learn in the situation of only few samples available from the desired distribution. The second is the ability to learn by using knowledge obtained from historical experienced tasks to quickly infer new knowledge or skills for the new task. To address these challenges, meta-learning was introduced as an efficient method to learn with very minimal input data by using inference from old tasks in the same task distribution.

In general, few-shot meta-learning technique is assumed to have access to a distribution of tasks and speedily adjust its networks to the new task in the distribution using merely few samples of this task and also a few update iterations. It can be done by encoding the neural learner system in weights of a recurrent network as stated in [1] and [2]. This approach predicts the new information with the use of additional architecture of recurrent models such as LSTMs and augmented memory by feeding sequential data to it. The other approach of meta learning is to build a generally optimal initialization to a variety of tasks under a distribution and start learning from this initial point for quickly mastering the new task. A typical work, [3], proposes a model-agnostic meta-learning (MAML) algorithm which gains initial network parameters by training a meta learner with K examples for each sampled task across a task distribution. The corresponding gradient which directs the initial network parameter $\emptyset$ to a specific task is then normalized over a whole set of sampled tasks to update the meta learner. As stated in [4], the performance of the initialization method such as [3] outweighs the former with RNN-based approach.

Introduced as an algorithm closely similar but with slightly better performance than First-Order MAML algorithm, Reptile in [4] introduces a simple calculation to update meta learner in which the gradient amount added to $\emptyset$ is substituted by the simple subtraction of $\widetilde{\emptyset}_i$, the adjusted parameter of meta learner to the training task i after k updates, and $\emptyset$ as the following formulas

$$\widetilde{\emptyset}_i = \emptyset + g_1 + g_2 + \cdots + g_k$$

$$\emptyset \leftarrow \emptyset + \epsilon \frac{1}{n} \sum_{i=1}^{n} (\widetilde{\emptyset}_\iota - \emptyset)$$

, where g_j with $j = 1, \ldots, k$ is the gradient of meta leaner to task i.

Reptile shows its effectiveness in few-shot regression and few-shot classification, however, its application to reinforcement learning is still showing undesirable result [4]. Aiming to prove the possibility of Reptile to the latter domain, the authors will propose a meta-learning implementation of Reptile to a classic Reinforcement Learning environment: Pendulum. Some evaluations and initial results comparing between Reptile and Random (or Xavier) Initialization will be presented in this paper for this purpose.

2. REPTILE ALGORITHM AND ITS IMPLEMENTATION WITH DEEP REINFORCEMENT LEARNING ALGORITHM

2.1. Reptile Algorithm

Reptile focuses on the minimization problem for the loss gaining from various tasks [4].

$$minimize_{\emptyset}\, E_\tau\left[L_\tau\left(U_\tau^k(\emptyset)\right)\right]$$

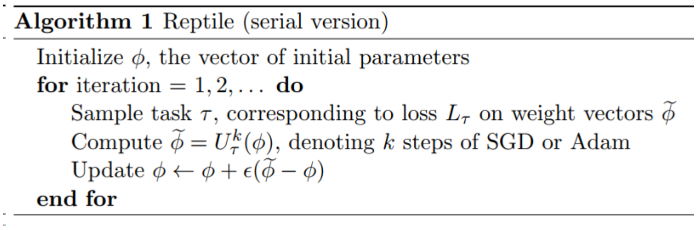
In the below formula, L_τ is the loss function for a sampled task τ and U_τ^k is the update operator $\emptyset$ of after k updates with training samples for task τ.

$$U_\tau^k(\emptyset) = \emptyset + g_1 + g_2 + \cdots + g_k$$

Reptile is a first-order meta-learning use a single update for $\emptyset$ with k = 1 and drive the update in the direction of simplified form of $\nabla_\emptyset L_\tau(\emptyset)$ as below.

$$g_{Reptile} = E_\tau[\nabla_\emptyset L_\tau(\emptyset)] = E_\tau[\emptyset - U_\tau(\emptyset)]/\alpha \quad (1)$$

The resulted algorithm from equation (1) is proposed in [4] and cited as Algorithm 1 – Reptile (Serial version).



Algorithm 1 Reptile (serial version)

```
Initialize $\phi$, the vector of initial parameters
for iteration = $1, 2, \ldots$ do
    Sample task $\tau$, corresponding to loss $L_\tau$ on weight vectors $\widetilde{\phi}$
    Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting $k$ steps of SGD or Adam
    Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for
```

2.2. DDPG with Reptile Algorithm

Inherited from this general algorithm, the authors have combined it with Deep Deterministic Policy Gradient (DDPG) algorithm to apply this simple form of first-order meta-learning on Reinforcement Learning domain (Algorithm 2).

Algorithm 2 Reptile in DDPG Implementation

```
Initialize critic network $Q(s,a|\theta^Q)$, actor network $\mu(s|\theta^\mu)$, target network $Q'$ and $\mu'$ for meta-learning DDPG agent.
for i = 1, N do
    Sample an environment $\tau_i$.
    Duplicate $Q$, $\mu$, $Q'$, $\mu'$ for task agent $Q_i$, $\mu_i$, $Q_i'$, $\mu_i'$.
    Clear replay buffer and re-initialize it with random exploration
        action in $\tau_i$.
    for episode = 1, M do
        Initialize Ornstein-Uhlenbeck noise $N_{OU}$ for action
            Exploration.
        Receive initial observation $s_1$ from environment reset.
        for t = 1, T do
            Perform action $a_t = \mu(s_t|\theta^\mu) + N_{OU}$, and store the
                transition $(s_t, a_t, r_t, s_{t+1})$ in R.
            Select a sampled minibatch from R and update $Q_i$, $\mu_i$,
                $Q_i'$, $\mu_i'$ according to DDPG.
        end for
    end for
    Update $Q$, $\mu$, $Q'$, $\mu'$ according to algorithm 1 with decaying
coefficient $\epsilon = \epsilon_0\left(1 - \frac{i}{N}\right)$.
```

3. RESULTS AND DISCUSSION:

3.1. Testing Environment Distribution

The task distribution selected for testing algorithm 2 is a continuous action domain: Pendulum-v0 from Open AI Gym library [5]. In this environment distribution, the length of pendulum is varied and each length setting is considered as a sampled task distribution.

The range of length is real number selected randomly in the range [0.5, 2] with the notification that DDPG with Reptile can only show its meaningfulness in the situation that normal DDPG (learning from random or

Xavier initialization) can learn optimal policy.

DDPG with Reptile's performance will be compare with DDPG without Reptile initialization to show the effectiveness of meta-learning in Reinforcement Learning domain.

3.2. Results and Discussion:

Testing DDPG with Reptile for few updates with 40 sampled environments shows a promising result for faster convergence with just fewer training episodes comparing with DDPG starting from scratch for new pendulum environment setting.

From figure 1, the effectiveness of Reptile cannot be expressed much in the case that meta learner is only updated with k = 1 task pre-trained (similar to joint training mention in [4]). Its performance is very similar to that of random initialization.

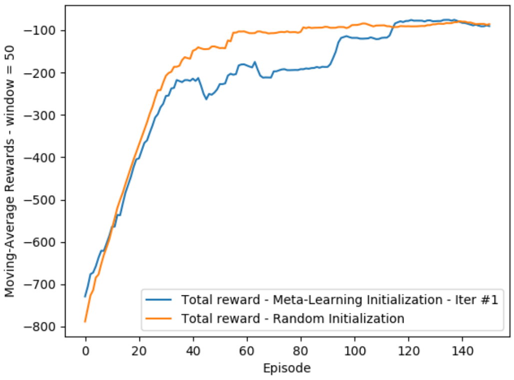


Figure 1. Total reward per episode for k = 1

As number of gradient update increases, DDPG with Reptile performance increased and show the power of first-order meta-learning to react to new task environment. Figure 2 and figure 3 are for the case of k varied from 1 to 25 and from 1 to 40. As can be seen from these figures, the more tasks experienced, the better meta-learner can anticipate and react to the new environment in testing phase.

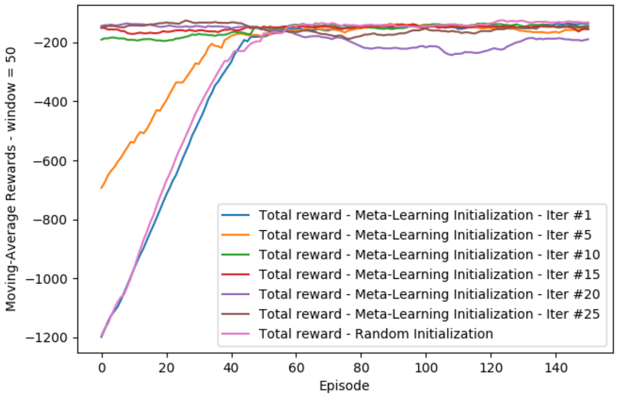


Figure 2. Total reward per episode for k = 1 to 25

4. CONCLUSION

This work has introduced an implementation addressed to the need of implementing Reptile to Reinforcement Learning domain as stated as future work in its original paper. The algorithm is combined with DDPG to handle Pendulum environment with continuous action domain for varied length of pendulum. Testing with random sampled environment with pendulum length in feasible range for DDPG to learn from 0.5 to 2, DDPG with Reptile algorithm has shown its effectiveness for very fast convergence and reaches to reasonable level of skillful policy after just few episodes training under new environment. This result has proved the possibility of applying the simple first-order meta-learning Reptile algorithm in Reinforcement Learning areas for few-shot learning problems.

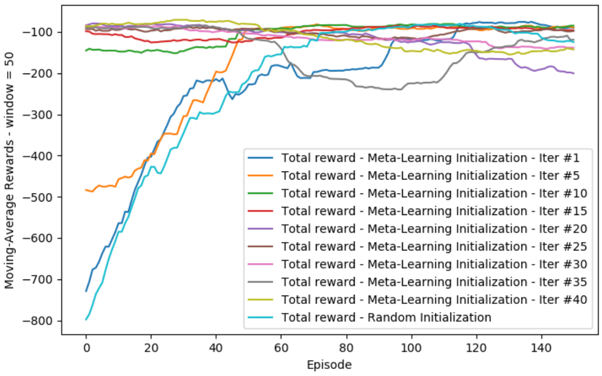


Figure 3. Total reward per episode for k = 1 to 40

ACKNOWLEDGMENT

The authors are grateful to the Basic Science Research Program through the National Research Foundation of Korea (NRF-2017R1D1A1B04036354).

REFERENCES

[1] S. Hochreiter, A. S. Younger and P. R. Conwell, "Learning to learn using gradient descent," in *International Conference on Artificial Neural Networks*, Vienna, Austria, 2001.

[2] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra and T. Lillicrap, "Meta-Learning with Memory-Augmented Neural Networks," in *The 33rd International Conference on Machine Learning* , New York City, NY, USA, 2016.

[3] C. Finn, P. Abbeel and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," in *The 34th International*

Conference on Machine Learning, Sydney, Australia, 2017.

[4] A. Nichol, J. Achiam and J. Schulman, "On First-Order Meta-Learning Algorithms," ArXiv, 2018.

[5] "Pendulum-v0," OpenAI, [Online]. Available: https://gym.openai.com/envs/Pendulum-v0/. [Accessed 2018].