*Code available at: https://github.com/duanwenbo/aaai22

Playing with Alchemy: A Benchmark and Evaluation for Meta-RL*

Wenbo Duan, Xiaoyang Wang
Department of Electrical and Electronic Engineering

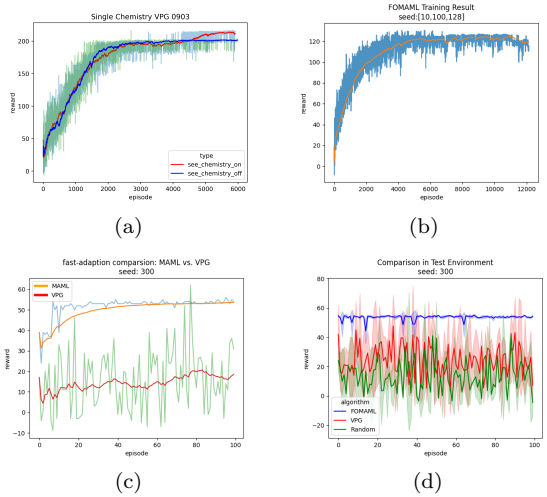
I. INTROCUTION

Meta reinforcement learning (meta-RL) has played an important role in the fast-adaptation of RL models. There haven't been many works on 'Alchemy', which is one of the few benchmarking platforms for meta-RL. Having implemented different RL algorithms on it, we concluded Alchemy as a challenging environment with complex latent structures, requiring high computational expense for evaluating meta-RL algorithms. By leveraging observations from studying the Alchemy, we design a customized environment with more flexible tasks. Three suggestions were proposed as ways to improve the effectiveness of the gradient-based meta-RL in subsequent experiments, with respect to robustness, computational expense and task distribution difficulties.

II. KEY RESULTS

Observations on the Alchemy Environment: Throughout trials of Vanilla Policy Gradient (VPG) and Proximal Policy Optimization (PPO) on a fixed 'chemistry' in Alchemy, we verified the effectiveness of the environment towards different types of reinforcement learning algorithms. The increasing and converging trends in Fig 1.a indicate the latent structure were being learnt by the agent.

However, meta-learning on this environment has been challenging. Regardless that Fig 1.b demonstrates the generalization ability of Model-Agnostic Meta-Learning (MAML) algorithm [1] in this environment, the final scores are far lower compared with the Oracle in [2]. According to our experiment, $1e5$ training episodes averagely takes about 26 hours, but an ideal training in the paper usually required the number of training steps in millions, which could take 100 times longer with our current computing resources.

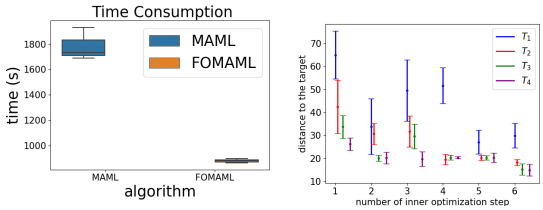


(a) (b)

(c) (d)

Figura 1. **a**: The single task performance validate the effectiveness of the algorithm and platform. **b**: The MAML result achieved a converge result but the final score is low. **c**,**d**: Two downside figures shows the effectiveness of the meta-learning compared with random, pre-trained algorithms in fine-tuning and testing scenarios.

Through our experiments, we conclude that despite the Alchemy achieves structural richness and structural transparency, the combination of 167,424 possible chemistries in the task distribution lead to a significantly difficult environment structure for the meta-RL agent to explore. Hence, benchmarking the performance of the algorithm in a non-industrial environment is not recommended.



(a) (b)

Figura 2. **a:** Time consumption of FOMAML is 40 % less than MAML. **b:** Average distance and standard deviation during training result. Increasing inner-optimization steps greatly increases the stability of training.

A Flexible Customized Environment: Inspired by issues on the Alchemy platform, we created a customized navigation environment which is inherited from the OpenAI gym library, designed with obstacles as well as the flexibility to switch between the obstacles distributions and task distributions. These flexibility made the environment a convenient tool to do fast verification and algorithms research.

Three Findings to Improve Training Performance: Based on the initial experiment result, we analyzed the factors that affect the performance of gradient-based meta reinforcement learning. Three aspects were found by exploring the customized environment. Multi-step inner optimization and the first-order approximation are verified with positive impact to some extend, while the variation of task distributions is found as a strong negative factor. As illustrated in Fig 2.a and 2.b, we concluded three useful suggestions for a more stable training:

A: Increasing the inner-optimization steps in gradient-based meta-RL algorithms is a way to balance between quick-adaptation and stable training, and hence, achieving a more robust model.

B: First Order MAML (FOMAML) can achieve a similar result with 40 % less time consumption than MAML.

C: Designing a difficulty classifier in task distribution is expected to lead to a better performance result.

[1] C. Finn, P. Abbeel, and S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in *ICML* (2017) pp. 1126–1135.

[2] J. X. Wang, M. King, N. Porcel, *et al.*, Alchemy: A structured task distribution for meta-reinforcement learning, arXiv preprint arXiv:2102.02926 (2021).

Towards Training Effectiveness in Gradient-Based Meta-Reinforcement Learning

Wenbo Duan[1*], **Xiaoyang Wang**[1]

[1]Department of Electrical and Electronic Engineering, University of Bristol
Bristol, UK, BS8 1UB
{pv19120, xiaoyang.wang}@bristol.ac.uk

Abstract

Gradient-based meta reinforcement learning algorithms like MAML are widely used in recent years. This paper analyzes the factors that affect the performance of gradient-based meta reinforcement learning, like the inner optimization steps and first-order derivative approximation through a customized environment with fully controlled difficulty. It demonstrates the way towards more robust training and better generalization.

Introduction

Reinforcement Learning (RL) methods have been demonstrated to be effective in a wide variety of sequential decision-making problems (Schrittwieser et al. 2020). The common bottlenecks of RL include low sample efficiency and poor generalization ability to unknown distributions. Meta reinforcement learning (Meta-RL) puts the learning object to a higher hierarchy, where it focuses on the internal links between a collection of structurally similar tasks, allowing the meta learner to explore and utilise latent knowledge shared by a set of tasks, and hence, quickly adapt to the new tasks. One classic family of algorithms for addressing Meta-RL problems are the gradient-based meta-learning algorithms, including Model Agnostic Meta-Learning (MAML) (Finn, Abbeel, and Levine 2017) and it's variants (Nichol, Achiam, and Schulman 2018). Concretely, MAML proposed a bi-loop optimization structure, in which the inner loop performs a one-step policy optimization across tasks sampled from distribution, while the outer loop seeks to find the best parameters for the meta-policy to fast adapt to a new context.

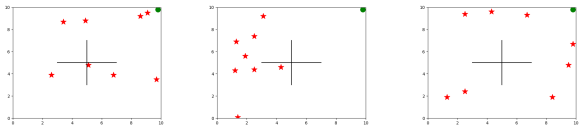
Despite being a simple and versatile method, MAML also faces difficulties like high computation expense and sometimes, poor robustness (Nguyen et al. 2021). In this work, we study the training in MAML through a customized environment, investigating factors that may affect the training effectiveness in three aspects:

- Robustness: The balance between quick adaptation and stable training.
- Computational expense: MAML and First order MAML (FOMAML)
- The effect of difficulties in task distributions.

*Student author. Tel: +447419903426

The Research Approach and Results

Customized Environment In meta-RL, we consider a distribution $p(\tau)$ over tasks, where each task τ_i is a different Markov decision process (MDP) M_i. Here $M_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_i, \rho_i, \gamma)$ represents the state space, action space, transition probability, reward, initial state distribution and the discount factor, respectively. To build the task distribution, we design a navigation environment with 3 target distributing schemes (Figure 1) as well as 4 obstacle distributing schemes, resulting in different levels of difficulties. We choose the environment with medium-level obstacle distribution and uniform target distribution as the base environment in the following study.



(a) Random (b) Narrow (c) Uniform

Figure 1: Task distributions with the "medium" level obstacles. The agent starts from the green points $(9.9, 9.9)$, aiming to find the optimal path to reach the red star (target). Each experiment comprises 8 target locations as a training batch for the meta learner.

Multi-steps Inner Optimization MAML comprises two layers of optimizations. In the inner loop, the policy network $f(\theta)$ is updated through one step of gradient descent $\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i} f(\theta)$ on each task τ_i. The outer loop takes the updated policy $f_{\theta_i^{'}}$ and backpropagate to the meta policy: $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{\tau_i}(f_{\theta_i^{'}})$. In this study, we analyze the number of steps taken in the inner loop, revealing the relationship between quick adaptation and training stability over the given task distribution.

We focus on the first 400 training episodes, analyzing the impact of different inner gradient steps N. In standard MAML, $N = 1$. Here N varies between 1 to 6. The 400 training episodes are divided into 4 successive time frames

$T_i, i = [1, ..., 4]$. Here T_1 represents episode 1 to 100, T_2 represents episode 101 to 200, and so on. Figure 2 shows the average distance and standard deviation during training from T_1 to T_4, with different number of inner gradient steps. Obviously, the meta-policy learns to solve the navigation problem regardless of N. More importantly, with the increase of N, the standard deviation between successive time frames is reduced distinctively, suggesting more stable training. Intuitively, increasing N gradually moves the policy away from the training task distribution, i.e., more gradient steps are required to adapt to new tasks. However, with N being a reasonable value, the meta-policy can benefit from both quick adaptation and stabilized training, achieving a good balance. Considering that one-step adaptation is not always feasible for complicated or shifted task distributions (e.g., the sim-to-real problem), our results indicate a way to fine-tune MAML to improve its stability while retaining the model adaptation ability to new tasks.

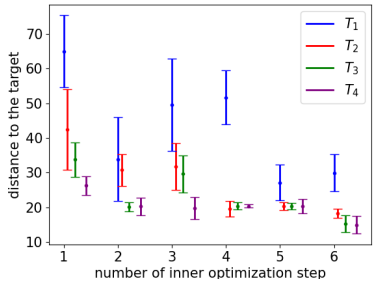
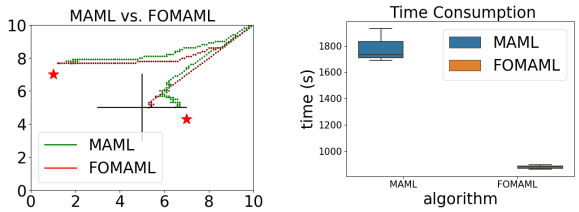


Figure 2: Multi-step inner-optimization. Comparison between the mean value and standard deviation when N varies between 1 to 6, within 4 successive time frames T_1 to T_4. The standard deviation at the same time frame indicates the training stability under different values of N.

First Order MAML In MAML, the computational cost brought by the second-order derivative is an obstacle in the algorithm application. First-order alternatives have been proposed, approximating the second-order derivative by first-order operations (Nichol, Achiam, and Schulman 2018). In this study, we focus on the First Order MAML (FOMAML) algorithms. We investigate the effectiveness of FOMAML in our customized environment comparing with the original MAML algorithm. As illustrated in Figure 3, the trained MAML and FOMAML models present very similar performance in our testing tasks, while the training speed of FOMAML accelerates by about 56% on average. This would be of significant help in the algorithm deployment.

Despite the huge improvement in FOMAML's training efficiency, our experiment indicates that when the perceptual features are relatively complicated, both algorithms are still prone to trap in a local minimum. As shown in Figure 3(a), MAML and FOMAML both fail to recognise or bypass obstacles in the centre of the map when trying to approach the bottom right target in a limited time (2000 episodes of training in the environments shown in Figure 1(c)). In the next section, we discuss the task distribution and difficulty, and a



(a) MAML vs. FOMAML. (b) Time Consumption

Figure 3: A comparison between MAML and FOMAML. (a): A visualisation of trajectories obtained from MAML and FOMAML agents, approaching two different targets. (b): Time consumption comparison.

potential way to improve gradient-based meta-RL.

Task Distributions and Difficulties [In progress] Figure 3(a) reveals an intuitive relation between latent environment difficulty and training efficacy. Recent research proposes to apply curriculum learning in the gradient-based Meta-RL (Mehta et al. 2020), which holds the idea of learning from the simple to the hard to solve various task distributions. The task distribution and difficulties can be fully controlled by 3 types of target distribution and 4 types of obstacles. This in-progress work of investigating the training of gradient-based meta-RL with clearly defined difficulties could bring insightful information on the robustness and generalization ability.

Conclusion

We analyze the factors that affect the effectiveness of gradient-based meta-learning algorithms through a customized environment. Multi-step inner optimization and the first-order approximation are verified with positive impact to some extend, while the variation of task distributions is found as a strong negative factor.

References

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 1126–1135. PMLR.

Mehta, B.; Deleu, T.; Raparthy, S. C.; Pal, C. J.; and Paull, L. 2020. Curriculum in gradient-based meta-reinforcement learning. *arXiv preprint arXiv:2002.07956*.

Nguyen, T.; Luu, T.; Pham, T.; Rakhimkul, S.; and Yoo, C. D. 2021. Robust MAML: Prioritization task buffer with adaptive learning process for model-agnostic meta-learning. In *ICASSP 2021*, 3460–3464. IEEE.

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Supplementary Materials

Code Availability

The source code, customized environment with different difficulties as well as trained models are available at `https://github.com/duanwenbo/aaai22`.

The Environment

The customized navigation environment is inherited from the OpenAI gym library, designed with the flexibility to switch between the obstacles distributions and task distributions. In terms of a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_i, \gamma)$, the action space $\mathcal{A}$ is discrete, representing 8 directions. The state representation in $\mathcal{S}$ has two versions: 1) the current position of the agent; 2) the current position of the agent along with the target information. This is inspired by a previous work (Wang et al. 2021), which mentioned the idea of exposing some latent information to the agent to accelerate training. The environment is designed with 3 task distributions and 4 obstacles distributions including null obstacles (see Figure 2). Each configuration has a clear difficulty distinction from other configurations. The reward is the negative square of the distance between the agent and the target:

$$\mathcal{R}_i = -\{(\mathcal{S}_x^i - \mathcal{G}_x)^2 + (\mathcal{S}_y^i - \mathcal{G}_y)^2\} \tag{1}$$

where $\mathcal{G}$ is the target position. In evaluation, we use the distance from the agent to the target, presenting intuitive results.

Experiments

Baseline Experiment Prior to future experiments, a reference experiment should be conducted as the baseline. As a baseline, we require a responsive environment as well as a consistent capability performance. Experiments with three distinct configurations were performed. As illustrated in Figure1, we investigated both the environment with and without barriers, as well as the effect of having more observation information during training (Wang et al. 2021). The environment with Medium obstacles and the observation with additional information is chosen as the baseline experiment setting since it is more stable than the experiment without additional information, which can be further adjusted.

Multi-step Inner Loop Optimization The parameters setting for the multi-step inner optimization is listed in the Table 1.

Our experiments show that multi-step inner optimization is a practical solution to increase training stability in Meta-RL. Figure 3(a) shows the comparison between 1, 3, 4 and 5 inner optimization steps, with 3000 training episodes in total, where Figure 3(b) shows the first 100 training episodes. It can be seen that these results are in alignment with the conclusion in our main abstract, as the standard deviation shrinks sharply when the inner optimization step increases. In Figure 3(a), we can also see that MAML algorithms with different inner optimization steps show similar performance at the end of training.

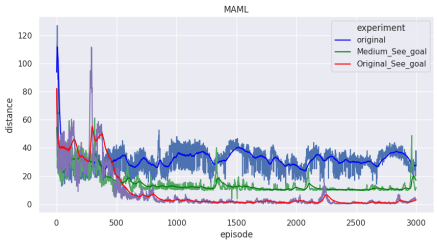


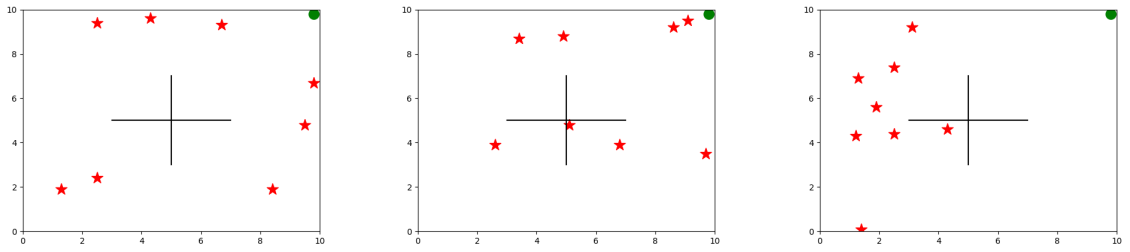
Figure 1: Training MAML with different environment configurations. From the top to bottom the line represents Blue: the MAML in an ordinary navigation environment, the input state is the current location; Green: the MAML in a navigation environment with a "Medium" obstacles distribution, the input is the current location as well as the target location; Purple: the MAML in an ordinary navigation environment, the input is the current location as well as the target location

Parameters	
Name	Setting
Inner-loop policy optimization algorithm	Vanilla Policy Gradient
Inner-optimizer learning rate	0.0003
Outer-optimizer learning rate	0.0005
Task distribution	Uniform
Obstacles distribution	Medium
Inner-loop tasks batch size	8
Training episode	3000
Hidden layers of convolutional neural network	128

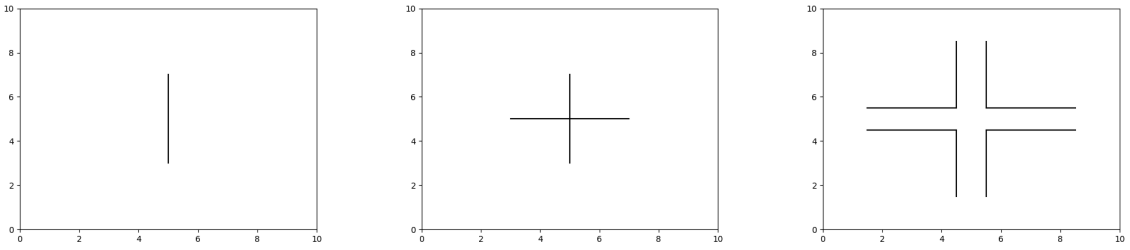
Table 1: Parameters setting for the multi-step inner loop optimization experiment.

This might bring up the doubt for the practical value of this method. We provide two aspects of justifications. 1) Lower standard deviation means more stable training and less sensitive to the model parameter initialisation, which is always desirable in practice. 2) Multi-step inner optimization can slightly relieve the dependency between meta-policy and the training task distribution while retaining the quick adaptation ability. This can be further used in distribution shifting scenarios. One example is the sim-to-real problem, where a gap exists when transferring trained model to real-world applications. Some recent research studies the distribution shift problem, pointing out the difficulty to migrate the model which trained well in the simulation environment to the practical scenarios (Zhao, Queralta, and Westerlund 2020). One of the factors that affect the model transferring is the serious bias on transition probability from the simulation to the real environment. Throughout the result of our experiment, we could leverage the idea of increasing the inner optimization steps. By slightly reducing the dependency between meta-policy and the training task distribution, the model transfers more general knowledge about the task, which generalizes better when facing out-of-distribution tasks and distribution shifts.

First Order MAML Experiments on FOMAML also use the parameters in Table 1. In optimization, Adam optimizer is used in FOMAML while MAML uses stochastic gradient descent (SGD) optimizer.



(a) Uniform target Distribution (b) Random target Distribution (c) Narrow target Distribution



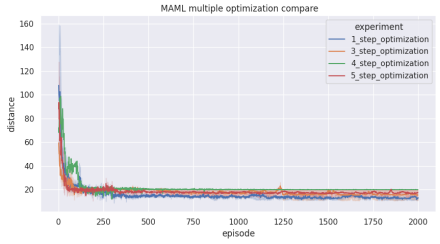
(d) Easy Obstacles Distribution (e) Medium Obstacles Distribution (f) Hard Obstacles Distribution

Figure 2: Our customized environment offers flexible tuning in two aspects and each aspect with three options. A batch of tasks could be uniformly, randomly, or narrowly distributed in the environment to test the algorithm's environment generalization ability in various difficulty levels. Obstacles in the environment are also designed as three different levels, higher level means a more difficult latent state structure for the meta algorithms.

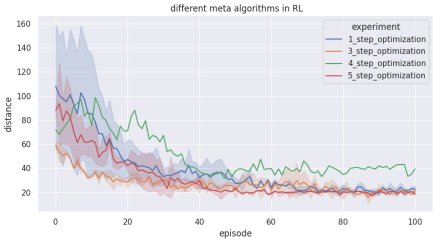
References

Wang, J. X.; King, M.; Porcel, N.; Kurth-Nelson, Z.; Zhu, T.; Deck, C.; Choy, P.; Cassin, M.; Reynolds, M.; Song, F.; et al. 2021. Alchemy: A structured task distribution for meta-reinforcement learning. *arXiv preprint arXiv:2102.02926*.

Zhao, W.; Queralta, J. P.; and Westerlund, T. 2020. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 737–744. IEEE.



(a) Training with 3000 episodes.



(b) The first 100 training episodes.

Figure 3: Overall performance of multiple inner loop optimization algorithm