MGHRL: META GOAL-GENERATION FOR HIERARCHICAL REINFORCEMENT LEARNING

Haotian Fu[1], **Hongyao Tang**[1], **Jianye Hao**[1], **Wulong Liu**[2], **Chen Chen**[2]
{haotianfu, bluecontra, jianye.hao}@tju.edu.cn, {liuwulong, chenchen9}@huawei.com
[1]Tianjin University, [2]Noah's Ark Lab, Huawei

ABSTRACT

Most meta reinforcement learning (meta-RL) methods learn to adapt to new tasks by directly optimizing the parameters of policies over primitive action space. Such algorithms work well in tasks with relatively slight difference. However, when the task distribution becomes wider, it would be quite inefficient to directly learn such a meta-policy. In this paper, we propose a new meta-RL algorithm called Meta Goal-generation for Hierarchical RL (MGHRL). Instead of directly generating policies over primitive action space for new tasks, MGHRL learns to generate high-level meta strategies over subgoals given past experience and leaves the rest of how to achieve subgoals as independent RL subtasks. Our empirical results on several challenging simulated robotics environments show that our method enables more efficient and generalized meta-learning from past experience.

1 INTRODUCTION

Human intelligence is remarkable for their fast adaptation to many new situations using the knowledge learned from past experience. However, agents trained by conventional Deep Reinforcement Learning (DRL) methods (Mnih et al., 2015; Levine et al., 2016; Bengio & LeCun, 2016) can only learn one separate policy per task, failing to generalize to new tasks without additional large amount of training data. Meta reinforcement learning (Finn et al., 2017; Mishra et al., 2018; Duan et al., 2016) addresses such problems by learning how to learn. Given a number of tasks with similar structures, meta-RL methods enable agents learn such structure from previous experience on many tasks. Thus when encountering a new task, agents can quickly adapt to it with only a small amount of interactions.

Most current meta-RL methods leverage experience from previous tasks to adapt to new tasks by directly learning the policy parameters over primitive action space (Finn et al., 2017; Rakelly et al., 2019). Such approaches suffer from two problems: (i) For complex tasks which require sophisticated control strategies, it would be quite inefficient to directly learn such policy with one nonlinear function approximator and the adaptation to new tasks is prone to be inaccurate. This problem can become more severe in sparse reward settings. (ii) Current meta-RL methods focus on tasks with narrow distribution, how to generalize to new tasks with much more difference remains a problem.

To tackle such problems, we propose a new hierarchical meta-RL method that meta-learns high-level goal generation and leaves the learning of low-level policy for independent RL. Intuitively, this is quite similar to how a human being behaves: we usually transfer the overall understanding of similar tasks rather than remember specific actions. Our meta goal-generation framework is built on top of the architecture of PEARL (Rakelly et al., 2019) and a two level hierarchy inspired by HAC (Levy et al., 2019). Our evaluation on several simulated robotics tasks (Plappert et al., 2018) as well as some human-engineered wider-distribution tasks shows the superiority of MGHRL to state-of-the-art meta-RL method.

2 PRELIMINARIES

In our meta learning scenario, we assume a distribution of tasks $p(\tau)$ that we want our model to adapt to. Each task correspond to a different Markov Decision Process (MDP), $M_i = \{S, A, T_i, R_i\}$, with

state space S, action space A, transition distribution T_i, and reward function R_i. We assume that the transitions and reward function vary across tasks. Meta-RL aims to learn a policy that can adapt to maximize the expected reward for novel tasks from $p(\tau)$ as efficiently as possible.

PEARL (Rakelly et al., 2019) is an off-policy meta-reinforcement learning method that drastically improves sample efficiency comparing to previous meta-RL algorithms. The meta-training process of PEARL learns a policy that adapts to the task at hand by conditioning the history of past transitions, which we refer to as context c. Specifically, for the ith transition in task τ, $c_i^\tau = (s_i, a_i, r_i, s_i')$. PEARL leverages an inference network $q_\phi(z|c)$ and outputs probabilistic latent variable z. The parameters of $q(z|c)$ are optimized jointly with the parameters of the actor $\pi_\theta(a|s, z)$ and critic $Q_\theta^h(s, a, z)$, using the reparametrization trick (Kingma & Welling, 2014) to compute gradients for parameters of $q_\phi(z|c)$ through sampled probabilistic latent variable z.

3 ALGORITHM

3.1 TWO-LEVEL HIERARCHY

We set up a two-level hierarchical RL structure similar to HAC. As shown in Figure 1, High level policy μ^h takes in state and outputs subgoals at intervals. Low level policy μ^l takes in state and desired subgoals to generate primitive actions. Here, low level policy has at most K attempts of primitive action to achieve the desired subgoal, where K can be viewed as the maximum horizon of a subgoal action is a hyperparameter given by the user. As long as the low level policy μ^l run out of K attempts or the desired subgoal is achieved, this high level transition terminates. The high level policy uses agent's current state as the new observation and produced another subgoal for low level policy to achieve.

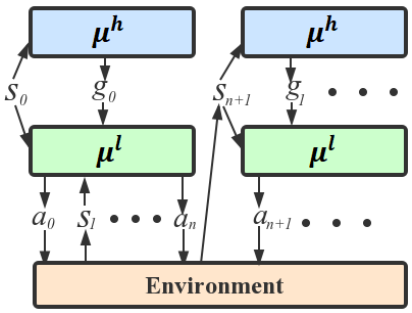

Figure 1: Two level Hierarchy

We use an a binary reward function for low level policy learning in which a reward of 0 is granted only if the subgoal produced by high level policy is achieved and a reward of -1 otherwise. Note that the environment's return (i.e. whether the agent successfully accomplished the task) will not affect the reward received by the low level policy. In our evaluation on simulated robotics environments, we use the positional features of the observations as the representation for subgoals. A subgoal is judged to be achieved only if the distance between subgoal and the gripper's current position s_{n+1} is less than threshold l.

3.2 META GOAL-GENERATION FOR HIERARCHICAL REINFORCEMENT LEARNING

The primary motivation for our hierarchical meta reinforcement learning strategy is that, when people try to solve new tasks using prior experience, they usually focus on the overall strategy we used in previous tasks instead of the primitive action execution mechanism. For instance, when we try to use the knowledge learned from riding bicycle to accelerate learning for riding motorcycle, the primitive action execution mechanism is entirely different although they share a similar high-level strategy (e.g. learn how to keep balance first). Thus, we take advantage of our two-level hierarchy structure and propose a new meta-RL framework called meta goal-generation for hierarchical RL (MGHRL). Instead of learning to generate detailed strategy for new tasks, MGHRL learns to generate overall strategy (subgoals) given past experience and leaves the detailed method of how to achieve the subgoals for independent RL. We leverage PEARL framework (Rakelly et al., 2019) to independently train a high level meta-policy which is able to quickly adapt to new tasks and generate proper subgoals. Note that off-policy RL method is indispensable in our structure when training high level policy due to its excellent sample efficiency during meta-training. And its structured exploration by reasoning about uncertainty over tasks is crucial to hierarchical parallel training framework. We leave the low level policy to be trained independently with non-meta RL algorithm using hindsight experience replay mechanism (Andrychowicz et al., 2017). In our simulated robotics experiments, the low level policy aims to move the gripper to the desired subgoal position which can be reused when switching to other tasks. Thus we only need to train a single set of low-level polices

which can be shared across different tasks. In other situations where the tasks are from different domains, we can choose to train low level policy independently on new tasks without using past experience.

We summarize our meta-training procedure in Algorithm 1. For each training task drawn from task distribution, we sample context c_h and generate hindsight transitions[1] for both levels of hierarchy ($line\ 4 \sim 13$) by performing current policy. Then we train high level and low level networks with the collected data ($line\ 16 \sim 22$).

Algorithm 1 MGHRL Meta-training

Require: Batch of training tasks $\{\tau_i\}_{i=1,...,T}$ from $p(\tau)$, maximum horizon K of subgoal action

1: Initialize replay buffers $\mathcal{B}_h^i$,$\mathcal{B}_l^i$ for each training task
2: **while** not done **do**
3: **for** each task τ_i **do**
4: Initialize high-level context $c_h^i = \{\}$
5: **for** m=1,...,M **do**
6: Sample $z \sim q_\phi(z|c_h^i)$
7: $g_i \leftarrow \mu_h(g|s,z)$
8: **for** K attempts or until g_i achieved **do**
9: Gather data using $a_i \leftarrow \mu_l(a|s,g)$
10: Generate hindsight action transition, hindsight goal transition and add to $\mathcal{B}_l^i$
11: **end for**
12: Generate hindsight transitions, subgoal test transitions and add to $\mathcal{B}_h^i$
13: Sample high level context $c_h^i = \{s_j, g_j, r_j, s'_j\}_{j=1,...,N} \sim \mathcal{B}_h^i$
14: **end for**
15: **end for**
16: **for** each training step **do**
17: **for** each task τ_i **do**
18: Sample high level context $c_h^i \sim \mathcal{B}_h^i$ and RL batch $b_h^i \sim \mathcal{B}_h^i$, $b_l^i \sim \mathcal{B}_l^i$
19: Sample $z \sim q_\phi(z|c_h^i)$ and calculate $L^h_{actor}(b_h^i, z)$, $L^h_{critic}(b_h^i, z)$, L^h_{KL}
20: Update low level actor and critic network with b_l^i
21: **end for**
22: Update high level networks with $\sum_i L^h_{actor}$, $\sum_i L^h_{critic}$, $\sum_i L^h_{KL}$
23: **end for**
24: **end while**

4 EXPERIMENTS

We evaluated our algorithm on several challenging continuous control robotics tasks (integrated with OpenAI Gym), simulated via the MuJoCo physics simulator (Todorov et al., 2012):

Fetch-Reach Fetch has to move the gripper to the desired goal position.

Fetch-Push Fetch has to move a box by pushing it until it reaches a desired goal position.

Fetch-Slide Fetch has to hit a puck across a long table such that it slides to rest on the desired goal.

Fetch-PickandPlace Fetch has to pick up a box from a table using its gripper and move it to a desired goal located on the table. To make exploration easier we recorded a single state in which the gripper is a few distance right above the box and start the training episodes from this state.

We compare our algorithm to baselines including PEARL with dense reward, HER-PEARL with sparse reward and HAC with shared policy. Note that Rakelly et al. (2019) has already shown that PEARL greatly outperforms other existing meta-RL methods like MAML (Finn et al., 2017), ProMP (Rothfuss et al., 2019) at both sample efficiency and final performance. Thus we mainly compare our results with it. In sparse reward setting, we further modify PEARL with Hindsight

[1]To achieve parallel training for the two levels of our framework, we rewrite past experience transitions as hindsight action transitions, and supplement both levels with additional sets of transitions as was done in HAC.

Experience Replay (Andrychowicz et al., 2017) for a fair comparison[2]. The last one means we train a shared HAC policy jointly across all meta-train tasks sampled from the whole task distribution.

We first do the simplest meta-learning evaluation on each type of the four tasks. In each scenario, we evaluate on 50 meta-train tasks and 10 meta-test tasks, where the difference between each task is in the terminal goal position we want the box or gripper to reach as well as the initial positions. The results are shown in Table 1. In Fetch-reach environment which is very easy to learn as we mentioned before, the tested methods all reach a final performance of 100% success rate. Our method MGHRL outperforms the other three methods in Push and Slide scenarios, while PEARL with dense reward performs better in Pick-Place tasks. Our two-level hierarchy and hindsight transitions significantly decrease the difficulty of meta learning with sparse reward, and is able to learn efficiently under a fixed budget of environment interactions. HAC with shared policy lacks generalization ability and cannot always achieve good performance when tested on varied tasks as shown in our results.

Table 1: Average success rates over all tasks (meta-test)

Tasks	MGHRL	PEARL	HER-PEARL	HAC
Reach	100%	100%	100%	100%
Push	**76%**	61%	15%	41%
Slide	**36%**	5%	6%	23%
Pick-Place	92%	**98%**	47%	13%

We further evaluate our method on tasks with wider distribution. As shown in Figure 2, each scenario's meta-train and meta-test tasks are sampled from the original two or three types of tasks (e.g. 30 meta-train tasks from Push and 30 meta-train tasks from Slide). Our algorithm MGHRL generally achieves better performance and adapts to new task much more quickly in all four types of scenarios. Directly using PEARL to learn a meta-policy that considers both overall strategy and detailed execution mechanism would decrease prediction accuracy and sample efficiency in these wider-distribution tasks as shown empirically. It is better to decompose the meta training process and focus on goal-generation learning. In this way, our agent only needs to learn a meta-policy that gives the learning rules for learning how to generate proper subgoals. Moreover, under dense reward settings of these challenging tasks, the critic of PEARL has to approximate a highly non-linear function that has different meanings for the two or three different types of tasks. Using the sparse return is much simpler since the critic only has to differentiate between successful and failed states.

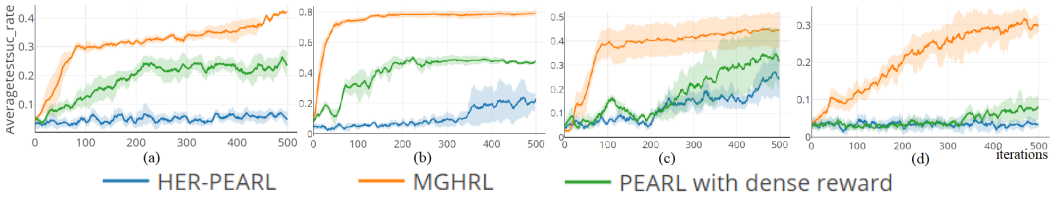

Figure 2: Average success rates for MGHRL, PEARL agents in each scenario: (a) Push & Slide & Pick-Place, (b) Push & Pick-Place, (c) Pick-Place & Slide, (d) Push & Slide. Each algorithm was trained for 1e6 steps. The error bar shows 1 standard deviation.

5 Discussion and Future Work

In this paper, we propose a hierarchical meta-RL algorithm, MGHRL, which realizes meta goal-generation and leaves the low-level policy for independent RL. MGHRL focuses on learning the overall strategy of tasks instead of learning detailed action execution mechanism to improve the efficiency and generality. Our experiments show that MGHRL outperforms the SOTA especially in problems with relatively wider task distribution. Beyond this paper, we believe our algorithm can accelerate the acquisition of entirely new tasks. For example, to learn tasks such as riding bicycles and riding a motorcycle, the two primitive action execution mechanism are entirely different but the two learning process still share similar high-level structures (e.g. how to keep balance). With meta learning on high level policy, our algorithm is still supposed to achieve good performance on such tasks. We leave these for future work to explore.

[2]We also evaluated PEARL (without HER) with sparse reward and it was not able to solve any of the tasks.

REFERENCES

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 5048–5058, 2017. URL http://papers.nips.cc/paper/7090-hindsight-experience-replay.

Yoshua Bengio and Yann LeCun (eds.). *4th International Conference on Learning Representations, ICLR 2016*, 2016. URL https://iclr.cc/archive/www/doku.php%3Fid=iclr2016:accepted-main.html.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl$ˆ2$: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016. URL http://arxiv.org/abs/1611.02779.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, pp. 1126–1135, 2017. URL http://proceedings.mlr.press/v70/finn17a.html.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014. URL http://arxiv.org/abs/1312.6114.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

Andrew Levy, George Konidaris, Robert Platt Jr., and Kate Saenko. Learning multi-level hierarchies with hindsight. In *7th International Conference on Learning Representations, ICLR 2019*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018*, 2018. URL https://openreview.net/forum?id=B1DmUzWAW.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018. URL http://arxiv.org/abs/1802.09464.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, pp. 5331–5340, 2019. URL http://proceedings.mlr.press/v97/rakelly19a.html.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. In *7th International Conference on Learning Representations, ICLR 2019*, 2019. URL https://openreview.net/forum?id=SkxXCi0qFX.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109.