Meta Reinforcement Learning with Generative Adversarial Reward from Expert Knowledge

Dongzi Wang
Computer School
National University of Defense Technology
Changsha, China
Dongzi91@qq.com

Bo Ding
Computer School
National University of Defense Technology
Changsha, China
dingbo@aliyun.com

Dawei Feng
Computer School
National University of Defense Technology
Changsha, China
davyfeng.c@qq.com

***Abstract*—Meta learning has been widely applied in the field of multi-task Reinforcement Learning. In meta-learning, a meta-model is obtained through a large number of pre-trainings and is able to adapt quickly and well on the unseen task in test. However, previous meta Reinforcement Learning methods often require vast computation cost and well-designed reward function, which are hardly available in many real world tasks, such as self-driving and robots control. On the other hand, in such cases there exist plenty of demonstrations from experts. In order to alleviate the above problems, this paper proposes a meta learning method with expert knowledge in sparse reward scenario. By introducing expert knowledge into a meta learning framework, the training speed and generalization performance of a meta-model are enhanced. Experiments show that our method can effectively improve the training speed of the meta-model. In addition, in sparse reward setting, the convergence speed, as well as the generalization ability of the proposed method, are significantly better than classic meta learning method.**

Keywords—Meta Learning, Multi-Task Reinforcement Learning, Imitation Learning, Inverse Reinforcement Learning

I. INTRODUCTION

Deep Reinforcement Learning has enabled agents to automatically explore and learn how to conduct pre-designated task effectively. However, in order to obtain a model capable of well adapting or generalizing to new tasks and new environments that have never been encountered during training time is of great challenge. To this essence, meta Reinforcement Learning (MRL), which aims to learn new concepts and skills fast with a few training examples has become a promising rescue.

In the MRL, the meta-model is trained over a variety of learning tasks and the adaptation is conducted during test with limited samples from the new task. To obtain a fair meta-model, a large number of training conducted on tasks with as much diversity as possible is required. Consequently, the meta-training process is always accompanied by vast computation cost and makes the convergence of the meta-model very slow. For example, in the model-agnostic meta-learning (MAML), agents have to explore a batch of sampled tasks during each episode in order to compute adapted parameters with gradient descent. Further, the number of sampled tasks has to be sufficiently large such that the diversity of training scenarios explored by agents is guaranteed. Thus, how to improve meta-training efficiency without losing final performance is a very challenging problem.

In addition, in most real world tasks rewards are sparse, which makes most Reinforcement Learning algorithms struggling with such sparsity. One alternative approach to this problem is to design reward function artificially beforehand so that rewards acquired by agents during task execution are dense, and the task becomes more suitable for learning. Generally, in some simple tasks, like CartPole, MountCar, LunarLander and some Atari games, the reward functions can be easily defined, and even the rewards are sparse in some scenarios, it is not difficult to shape the reward accordingly[1]. However, in most mechanical-control tasks, such as self-driving, auto-aircraft, robots control and other control-tasks in real world, their reward functions are generally hard to design manually. On the other hand, for these tasks, plenty of demonstrations from experts are available.

As mentioned above, the performance of meta-model depends on a well-defined reward function and feature extractor, which is difficult to obtain in many cases. Even if those two main elements is available, the large computation and time cost is unacceptable because aim of meta learning is to conclude a common policy for all task in a specific task domain, which is obviously harder than single task learning. Matthew make a point about the problem in a paper[2], they believe that "FAST" is from "SLOW" in meta-RL, and it is normal that learning is slow at the beginning of learning because agent need to explore large unseen state space and learn enough information about environment (This is meta-learning phase). After the slow "internship", agent get abundant common knowledge so that it can learn more fast in task domain (This is adapt phase). We totally agree with them, but we think there are a important difference between human learning and RL, that is when human start learning some skills there are always teachers or some professional books that can be consulted. So, human can quick get relevant knowledge and master skills after just little practice. In fact, there is much available expert knowledge in our world for RL, which can be contained in experts' brains or experts' demonstrations (expert can be a well-designed agent or human selves). So the main aim of this paper is to accelerate the meta-learning by introducing experts' knowledge into learning phase. And experts' knowledge also help us alleviate reward and feature design problem. The forms of Expert's knowledge can be various, such like: the rules of expert's behaviors, the parameters of model and the traces of experts, etc. In our paper, We default to expert knowledge as a behavioral trajectory because this form of knowledge is more readily available in reality.

978-1-7281-8304-6/20/$31.00 ©2020 IEEE

Dalian, China
September 27-29, 2020

In RL, although many Meta Learning methods have been proposed ([3, 4, 5]) recently, we focus on Model-agnostic meta-learning (MAML)[6] because it has no assumptions about environment dynamics and has brighter future in a variety of tasks. In practice, this paper tries to improve the meta-learning framework MAML by employing Imitation Learning (IL) based on expert samples. Specifically, we divide the training of a meta-model into two phases, namely "guidance" and "exploration". In the guidance phase, we introduce an Imitation Learning algorithm based on the generative adversarial neural network (GAIL) [7]. It integrates Supervised Learning and Inverse Reinforcement Learning and is suitable to acquire effective learning in sparse reward or hard-design reward scenarios. In this way, we make full use of the prior knowledge from experts and enable the meta-model to learn common knowledge in the task domain quickly and stably. In the second phase, i.e. the exploration phase, in order to improve the generalization ability of a meta-model, we use the exploration of sampling tasks to further train the meta-model, such that the meta-model's ability to master comprehensive knowledge is enhanced. Different from some methods that combined Imitation Learning with MAML, we don't abandon the original reward of tasks and we combine the Imitation and Reinforce in one learning framework so that our method have quick learning speed and good final performance.

To sum up, we propose a meta training method based on expert knowledge. By introducing expert knowledge into the MAML framework, we improve the training speed and generalization performance of a meta-model. Experiments show that our method can effectively improve the training speed of the meta-model under the normal reward setting. In the case of sparse reward setting, the training speed and generalization ability of the model are significantly better than the classic MAML method..

II. Preliminaries and Related Work

A. Preliminaries

A Markov Decision Process (MDP) $\mathcal{M}$ can be defined by a 5-tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{T}$, $\mathcal{I}$), where $\mathcal{S}$ is a state set and $\mathcal{A}$ is an action set. $\mathcal{R}$ (s, a) is an immediate reward of executing action a in s, and transition function $\mathcal{T}$ (s, a, s') is a probability of reaching state s' given that action a is taken in state s. $\mathcal{I}$ is a distribution of initial states. A policy π for $\mathcal{M}$ is a mapping from $\mathcal{S}$ to $\mathcal{A}$. The aim of Multi-Task learning is to find a unified policy π for agent so that it can solve a cluster of tasks simultaneously.

B. Multi-Task Reinforcement Learning

Traditionally, there are two main classes of method to solve multi-task RL learning. One is to use non-parametric Bayesian model, which shares knowledge between tasks [8]. Another is based on Policy Reuse [9], in which policies from previously learned tasks are reused with probability to guide the learning of new tasks. Deisenroth uses policy gradients to learn a single controller that is optimal on average overall training tasks[10]. Actor-mimic[11] is a typical Policy Reuse method, which exploits the use of deep reinforcement learning and model compression techniques to train a single policy network. Before methods try to find one constant model to solve every task. Those methods compress a set of task-specific policy $\pi_1, \pi_2 \cdots, \pi_N$ into one network. Because conventional methods want to get a policy that could solve Multi Markov Decision Process at once, that means there must exist a greater input and output space and the complexity of policy network increases very rapidly, those methods are hard to be scalable. What's more, those methods require that the best solution of every task must be close in parameter space, otherwise the performance is unpredictable. In some good case, where the diverse tasks have adequate latent similarity, the common policy work well. However, in some bad cases, where tasks much differ from each other in some respects, this algorithm shows very poor performance.

Meta learning opens up a new perspective in multi-task RL field. It pursues a compromise to every task. Meta Reinforcement Learning, or learning to learn, refers to the problem of learning policy which can adapt quickly to novel tasks by using prior experience on different but related tasks[12-18]. MAML is promising one of meta learning algorithms, and several method based on MAML has obtained impressive success[3, 19].

The parameters of a learned MAML model maybe not the best solution for some specific task, however, it is easy to achieve a promising result on a specific task by gradient optimization after a few steps adaptation. In test phase, MAML is able to improve the final performance in most cases. However, MAML shows an insufficient ability to solve the unstable learning situation, especially when the reward is sparse.

C. Multi-Task Imitation Learning

Imitation Learning (IL) drives much attention in recent years, especially in Multi-Task RL field. Behavioral cloning (BC) and Inverse Reinforcement Learning (IRL) are two typical methods in Imitation Learning. BC is a method that performs Supervised Learning (SL) from observations to actions (e.g. [20, 21]).

Behavioral cloning performs poorly on many complex tasks, because the distribution of demonstrations and that of agent's trajectories are usually different, which seriously obstructs the SL procedure. Dagger (Dataset Aggregation) [21] is a advanced BC method, and it mathematically guaranteed consistency of the distributions of training data and test data from perspective of SL by consulting expert to get good answers in agent exploring, but this algorithm need an expert which can be consulted at any time and much work is pushed into experts.(In most situations, this kind of expert is unavailable.) In Inverse Reinforcement Learning (IRL), a reward function [22-25] is estimated to explain the demonstrations as near-optimal behavior.

IRL is able to learn reward function form expert demonstrations, and further utilizes this learned reward function to train a policy network. Before, traditional IRL methods (e.g. [22, 23, 26, 27]) always need a good feature extractor, whose performance rely heavily on human experience. Ho and Ermon propose Generative Adversarial Imitation Learning (GAIL), which is inspired by GANs [28] and can learning reward function and policy End-to-End without manual feature. Ziyu and Josh combine the VAE[29] and GAIL to train a Multi-Task policy net, but this method can not be generalized to new task that agent haven't see in training phase in task domain.

Recently, some researchers also have introduced the main idea of meta-learning into Multi-Task Imitation Learning,

many of them combine the SL IL(like BC or DAgger) with MAML(e.g. [30-32]), which are hard to apply for a lot of scenarios because the mentioned defects of BC and DAgger. Adam and Oliver proposed a Multi-task Imitation Learning algorithm[33], they employ the MAML framework and use the policy network to fit the diverse expert demonstrations with the Maximum Entropy Inverse Reinforcement Learning Learning[23], in which a good feature extractor is must required. Different from above ways, our methods aims to accelerating the training speed by involving experts' knowledge.

III. METHOD

In order to get a meta-model that can quickly adapt to multi-task settings, we use the basic framework of MAML. And we import expert knowledge into meta-model by GAIL\cite{ho2016generative}.The difference lies in that an IRL is employed to update meta-model of MAML, which uses GANs framework to formulate a reward function for learning the similarity between agent's trajectories and expert's demonstrations. Training signals are provided via the learned reward function to guide the agent's learning process. In addition, we extend the method to complex assignments, like MuJoCo 3D physics control tasks. The algorithm is named as Meta-GAIL in the following section.

A. Meta Generative Adversarial Imitation Learning: Meta-GAIL

In order to get a meta-model that can quickly adapt to multi-task settings, we use the basic framework of MAML. And we import expert knowledge into meta-model by GAIL\cite{ho2016generative}.The difference lies in that an IRL is employed to update meta-model of MAML, which uses GANs framework to formulate a reward function for learning the similarity between agent's trajectories and expert's demonstrations. Training signals are provided via the learned reward function to guide the agent's learning process. In addition, we extend the method to complex assignments, like MuJoCo 3D physics control tasks. The algorithm is named as Meta-GAIL in the following section.

There are two phases (i.e. guidance phase with expert demonstrations and exploration phase) in the proposed method, as illustrated in Fig. 1, where each phase contains two loops for updating parameters.

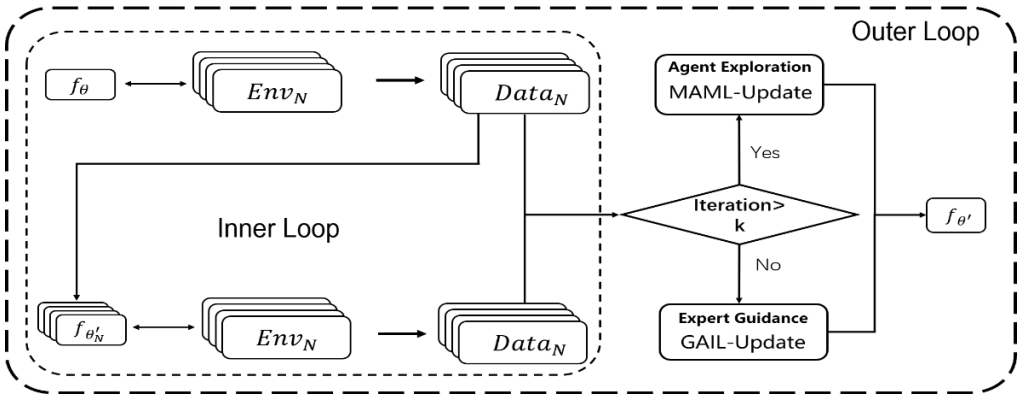

Fig. 1. Illustration of the Meta-GAIL framework. Meta-GAIL consists of two loops: in the inner loop, meta-model interacts with sampled tasks and generates a series of data; in the outer loop, we use these data to computes the gradient of parameters theta respect with Meta-loss by the PPO to update meta-model.

In the guidance phase, we expect to speed up the training process of the meta-model. Therefore, we introduce a fixed quantity of experts, each of which performs well on corresponding task. The meta-model is trained on those specific tasks with guidance from expert demonstration via policy gradient. In every iteration of the outer loop, the meta-model adapts one step on a specific task environment firstly. Then those policies are employed to interact with corresponding environments to obtain sample trajectories. Later, we use SL to train a neural network to evaluate the difference between those trajectories and the experts' demonstrations. In this way, a higher score is output for an action-state pair (s_t, a_t) that is more similar to the experts' demonstrations. Finally, above scores will be used as the reward signal to train the meta-model by Proximal Policy Optimization (PPO)[34] also. Because the number of experts is finite and discrete but the number of tasks in a task domain may be infinite, it's worth noting the way we chose the experts. Practically, we select the expert tasks uniformly form task domain, so that the task experts can cover the whole task domain as much as possible. In this way, the meta-model has more chances to learn more comprehensive knowledge about task domain.

In the exploration phase, the training procedure is the same as in MAML, except for two things: a) the training tasks are not fixed tasks same as above but sampled from distribution of tasks (the distribution is a uniform distribution in our experiment) at every outer iteration; b) the training reward is not from experts knowledge but the original reward which is given by task environment. The exploration phase is introduced to enable the meta-model to generalize well on the whole task domain.

In brief, we divide the training procedure into two phases, which are shown in **Algorithm 1**. In first stage (i.e. the guidance stage), we use experts' experience to guide agent to learn common knowledge of the task domain. During this stage, the training tasks are fixed and consistent with those tasks of expert knowledge.

Algorithm 1 Meta-GAIL

Require: $q(\mathcal{T})$: distribution over tasks
Require: $Dataset$: demonstrations of specific task experts
Require: k: the parameter that balance guidance and exploration
Require: α, β: step size hyperparameters
randomly initialize θ
Sample tasks batch $\mathcal{T}_i$ same as experts
while $iterations < k$ **do**
  Run **Algorithm 2**
  Update θ using PPO to minimize the objective in Equation (6) with step size β:
end while
while not done **do**
  Sample tasks batch $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample K trajectories $\mathcal{D} = \{(x_1, a_1, ...x_H)\}$ using f_θ in $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ by computing gradients of Equation (1)
    Compute adapted parameters θ_i with gradient descent: $\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample trajectories $\mathcal{D}_i = \{(x_1, a_1, ...x_H)\}$ using $f_{\theta_i^{'}}$ in $\mathcal{T}_i$
  end for
  Update θ using PPO to minimize the objective in Equation (3) with step size β
end while

More details are given in **Algorithm 2**.In the second stage (i.e. the exploration stage), we take the same training procedure as in MAML, where the training tasks in each iteration are sampled from the task domain.

Algorithm 2 GAIL-Update

Require: $\mathcal{T}_i$: a set of task same as experts
Require: $Discriminator_i$: a set of reward producers
Require: θ: meta parameter
for all $\mathcal{T}_i$ **do**
  Sample K trajectories $\mathcal{D} = \{(x_1, a_1, ...x_H)\}$ using f_θ in $\mathcal{T}_i$
  Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ by compute gradients of Equation (1)
  Compute adapted parameters θ_i with gradient descent: $\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  Sample trajectories $\mathcal{D}_i = \{(x_1, a_1, ...x_H)\}$ using $f_{\theta_i^{'}}$ in $\mathcal{T}_i$
end for
for all $Discriminator_i$ **do**
  Update $Discriminator_i$ using $Dataset_i$ and $\mathcal{D}_i$ by Equation (4)
  Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ by computing gradients of Equation (6)
end for
for all $\mathcal{T}_i$ **do**
  Sample K trajectories $\mathcal{D} = \{(x_1, a_1, ...x_H)\}$ using f_θ in $\mathcal{T}_i$
  Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ by computing gradients of Equation (1)
  Compute adapted parameters θ_i with gradient descent: $\theta_i^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  Sample trajectories $\mathcal{D}_i = \{(x_1, a_1, ...x_H)\}$ using $f_{\theta_i^{'}}$ in $\mathcal{T}_i$
  Replace corresponding rewards in $\mathcal{D}_i$ using the Equation (5)
end for

Besides, the number of available experts is always limited. Although experts' knowledge can accelerate the training process, the total amount of available experts is limited. As a result, expert knowledge can only cover a partial part of the task domain, which in other words, expert knowledge is incomplete, and sometimes is biased. As a result, in order to achieve a proper generalization in the task domain, we should also make good use of the exploration in addition to expert knowledge. Thus, how to balance guidance and exploration is one vital issue. In our experiment, we use a manual factor to balance the ratio between expert guidance and agent exploration.

Overall, in the guidance phase, Imitation Learning is introduced into the MAML framework to speed up agent training process with the utilization of expert knowledge. In the exploration phase, agent is enabled to explore more robust policies by using a conventional MAML parameters update. In practice, the proportion of utilizing experts' knowledge and self-exploration can be adjusted with a learnable threshold factor.

B. Training Objective

The meta-model is trained on different tasks simultaneously. As mentioned above, in the guidance phase, the training tasks are fixed, while in the exploration phase the training tasks are sampled from task domain. The loss of meta-model on task $\mathcal{T}$ can be formulated as following:

$$\mathcal{L}_{\mathcal{T}_i} = -\mathbb{E}_{x_i, a_t \sim f_\theta, q_{\mathcal{T}_i}} \left[\sum_{t=1}^{H} \mathcal{R}(x_t, a_t) \right], \quad (1)$$

Where θ is the meta parameters of model, and denotes $\mathcal{T}_i$ the task i. Each RL task $\mathcal{T}_i$ contains an initial state distribution $q_i(x_0)$ and a transition distribution $q_i = (x_{t+1}|x_t)$, and the loss $\mathcal{T}_i$ corresponds to the reward function $\mathcal{R}_i$. f_θ is a learned policy with parameters θ which maps states to actions. We optimize θ with Policy Gradient (PG) at this step.

The final aim is to minimize the overall loss on the batch-task cluster $\mathcal{T}$. With the loss of an agent obtained from specific task $\mathcal{T}_i$, the training objective can be defined as follows:

$$\mathcal{L}(\theta) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_\theta), \quad (2)$$

where $p(\mathcal{T})$ is task cluster $\mathcal{T}$ distribution, and $f_{\theta_i^{'}}$ are the adapted parameters of $\mathcal{T}_i$, and $f_{\theta_i^{'}}$ is the model of $\mathcal{T}_i$ after adaption. By combining Equation (1) and (2), the meta-objective can be defined as following:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T}_i)} \mathbb{E}_{x_t, a_t \sim f_{\theta'_i}, q_{\mathcal{T}_i}} \left[\sum_{t=1}^{H} \mathcal{R}_i(x_t, a_t) \right]. \quad (3)$$

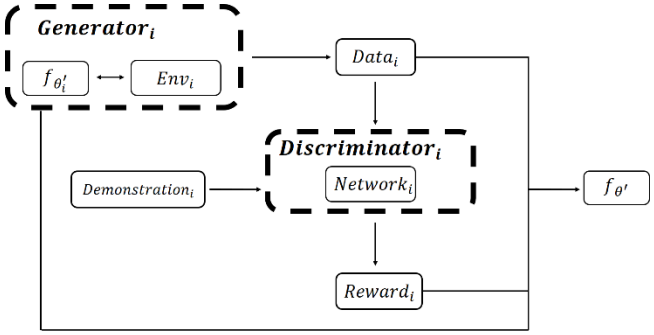

Fig. 2. Illustration of GAIL-update. $f_{\theta'_i}$ is the adapted model after agent adapts at task $\mathcal{T}_i$ and is the model $f_{\theta'}$ of new meta-model. In GAIL-update, agent interacts with Env_i with mdoel $f_{\theta'_i}$ and generates a series of trajectories $Data_i$, e.g. $(x_0, a_0, x_1, a_1, \ldots\ldots, x_N, a_N)$. The discriminator is a neural network, that decides whether each instance of trajectory that it reviews belongs to the actual expert demonstrations or not. The output discrepancy between generated trajectory and expert trajectory is provided as a training reward to the meta-model. Finally, parameters of the meta-model are computed by the PPO step.

In the case where a well-designed reward function $\mathcal{R}_i$ is hard to acquire, we use a rough reward function (i.e. an adversarial loss) as a replacement. As shown in Fig. 2, we use network D_i to denote the discriminator of GANs, which is used to discriminate an agent's trajectories from an expert's trajectories in $\mathcal{T}_i$. The generator G_i of GANs is naturally the policy network π, in which agent interact with environment and generates trajectories. In addition, we expect the policy does not contain any extra hypothesis about the task environment, therefore, we add entropy regularization to the optimization objective. Finally, the objective of this GANs in task $\mathcal{T}_i$ can be formulated as follows:

$$\min_{\pi_i}\max_{G_i}\mathcal{L}(D_i,\pi_i)=\mathbb{E}_{x_t,a_t\sim\pi_i}[logD_i(x_t,a_t)]+\mathbb{E}_{x_t,a_t\sim Data_i}[\log(1-D_i(x_t,a_t))]+\lambda H(\pi_i), \quad (4)$$

Where $H(\pi)=\mathbb{E}_\pi[-log\pi(a|s)]$ is γ-discounted causal entropy. And from (4) we can get a reward function:

$$\mathcal{R}(x_t,a_t)=logD_i(x_t,a_t). \quad (5)$$

As **Algorithm 2** shows, we update the policy π with PG in the inner iteration. Here, we replace the reward function in (3) with (5), the final objective turns into the following format:

$$\mathcal{L}(\theta)=-\mathbb{E}_{\mathcal{T}_i\sim p(\mathcal{T}_i)}\mathbb{E}_{x_t,a_t\sim f_{\theta'_i},q_{\mathcal{T}_i}}\left[\sum_{t=1}^{H}logD_i(x_t,a_t)\right], \quad (6)$$

IV. Experiment

A. Task Domain

In experiment, we use the Ant Walk as our task domain. Ant Walk is a standard continuous control task in MuJoCo. As the Fig. 3 shows, in Ant Walk, the agent can get an observation (125 dimensions vector) after it takes an action (8 dimensions vector). The aim of Ant Walk is to teach a quadruped robot to walk. In multi-task scenario, the aim of Ant Walk becomes more difficult, which is to teach the robot to walk forward at any specific velocity.

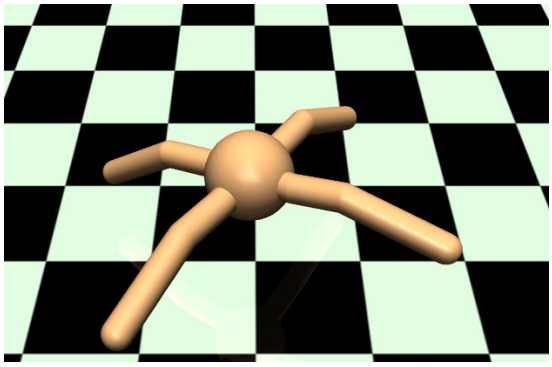

Fig. 3. Performance comparison of Meta-GAIL, MAML and Joint Training in dense reward setting.

All algorithms use the following parameters: the inner learning rate $\alpha = 0.01$, the outer learning rate , discount factor $\gamma = 0.99$, the horizon value $H = 200$. There are 20 rollouts per gradient step for each task. Policy network has identical hyperparameters: $input\ units = 125$, a LSTM[35] with two 100-units hidden layers, out units equals 8. The parameters k for balancing the guidance and exploration in Meta-GAIL is set to 65, which is chosen via experiments.

B. Dense Reward

In the setting of dense reward, reward mainly consists of three parts: the first part is used to evaluate the agent's own control ability; the second part reflects the agent's lifetime in each round; the third part measures the difference between the target speed and the agent's own actual speed, which is also our main focus. In our experiment, we randomly sample 40 new tasks at each step of training to evaluate the progress of training.

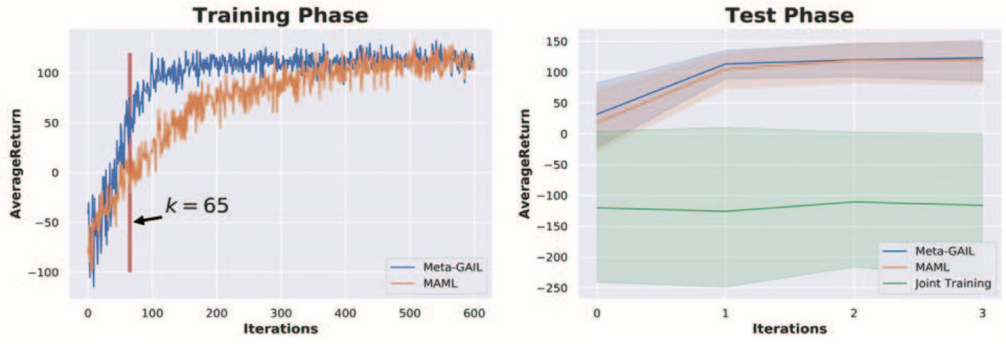

Fig. 4. Performance comparison of Meta-GAIL, MAML and Joint Training in dense reward setting.

Fig. 4 illustrates the performance comparison among Meta-GAIL, MAML, and Joint Training. The left sub-figure

shows the comparison between Meta-GAIL and MAML in the training process (results of Joint-training is not shown here, as it does not converge and severely affects the comparison of other experimental results).

It can be seen clearly that in the guidance phase (i.e. the first 65 steps), the learning speed of Meta-GAIL method is very fast. When iteration equals 65, the average return obtained by Meta-GAIL has reached around 50, while the return of MAML is around 0. When iteration exceeds 65, Meta-GAIL and MAML update the model in the same way (Both proceed training by exploration on sampled tasks). The learning speed(i.e. the slope of average return curve) shows a similar trend. As demonstrated, Meta-GAIL relies on a lot of prior knowledge accumulated in the guidance stage and converges faster in the exploration stage. On the other hand, due to the lack of expert guidance in the early stage, the MAML shows a very slow convergence. In terms of the final average return, no significant difference between Meta-GAIL and MAML is exhibited. However, Meta-GAIL only uses about 200 steps to complete the training, while MAML consumes about three times as much (about 600 steps).

The right sub-figure of Fig. 4 shows the performance of each model on test tasks after training. As can be seen from the figure, joint-training shows the worst performance, whose average return is about -130, and the variance is very large. We speculate that the main reason is that the number of iterations is too small for joint-training, through which JT's model is not able to learn common knowledge of the task domain. MAML and Meta-GAIL show similar performance, both can adapt well to the new tasks. Specifically, after four steps adaptation, the average return of both model reaches es about 125.

In the intensive reward setting experiment, we can clearly see that Meta-GAIL is able to accelerate the training process effectively. By learning from expert demonstration and exploring with the environments, Meta-GAIL can obtain a competitive generalization ability in the task domain.

C. Sparse Reward

In sparse reward setting, we increase the sparsity of reward by discretizing the reward return with a fixed value l (we set $l = 0.5$). Formally, the sparse reward R_t^s is defined as following:

$$R_t^s = \lfloor R_t / l \rfloor * l, \tag{7}$$

where R_t is the above dense reward of state-action pair (s_t, a_t).

In this setting, the reward signal an agent gets from the environment is very sparse. From the left sub-figure of Fig. 5, we can see that the MAML method demonstrates a great loss performance under the sparse reward setting. On the contrary, due to the guidance of expert knowledge, the performance of Meta-GAIL shows steady improvement, no drop is observed. In addition, when iteration equals 65, Meta-GAIL is able to achieve a good average return, approximately 40. While on MAML can only achieve about -25, a sharp drop compared with the dense reward setting. Further, when training iteration exceeds 65, Meta-GAIL shows a quick convergence, while the convergence of MAML exhibits a very slow increase, and it does not converge even at 600 steps. This suggests that in the case of sparse reward, it is more difficult for the MAML method to obtain continuous and direct reward feedback, which makes it rather difficult for agents to acquire useful knowledge about correctly choosing actions in the exploration. As a consequence, a significant drop in exploration efficiency is observed.

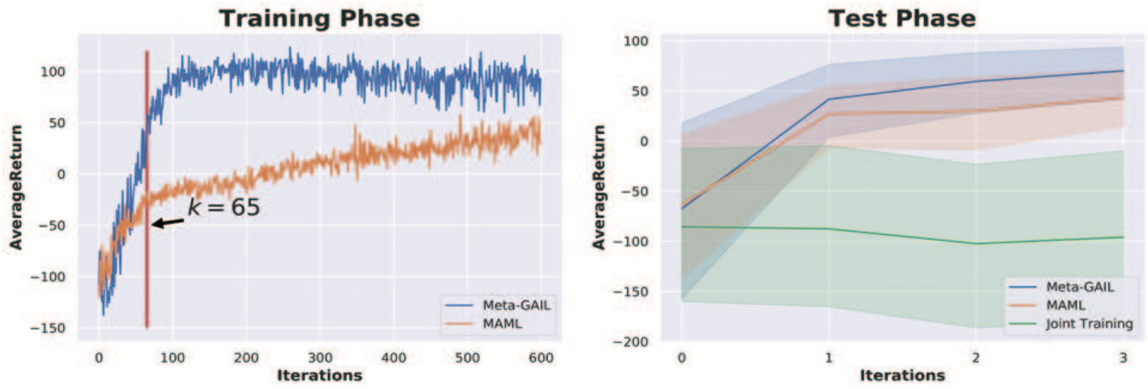

Fig. 5. Performance comparison of Meta-GAIL, MAML and Joint Training in sparse reward setting.

The right sub-figure of Fig. 5 shows the performance comparison among methods in the test phase. The joint training method still performs the worst. The difference is that after an adaptation at $k = 65$, Meta-GAIL demonstrates a better result than MAML. In addition, in training phase, we can see that after the iteration number exceeds 400, the performance of Meta-GAIL shows a weak decrease and its variance increases. We guess that with the training proceeds, the sparse reward signal drives the exploration process into an unstable state.

The above experimental results demonstrate the advantages of Meta-GAIL compared with other methods. In the case of sparse reward, Meta-GAIL still has excellent performance.

V. Discussion and Conclusion

To sum up, we propose a multi-task RL learning algorithm Meta-GAIL in this work. In order to tackle the sparse reward problem, Meta-GAIL transfers experts' knowledge to agent through Imitation Learning to help agent learn common knowledge of task domain. Experiment results demonstrate that Meta-GAIL can speed up the training process of meta-learning, and is able to generalize well on new tasks.

This work introduces an initial step towards meta learning with expert knowledge in sparse reward scenario. In the current approach, we use a manually set ratio to balance the

proportion of guidance from experts and exploration from agents in training. Actually, the proportion ratio could be learned from experiments, with the progress of training. Further, how to apply current approach to new fields such as multi-agent RL[36,37] is also a promising direction.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (No.61751208) and the Advanced Research Program (No.41412050202).

REFERENCES

[1] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping." pp. 278-287.

[2] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow," Trends in cognitive sciences, 2019.

[3] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies." pp. 5302-5311.

[4] J. X. Wang, Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J. Z. Leibo, D. Hassabis, and M. Botvinick, "Prefrontal cortex as a meta-reinforcement learning system," Nature neuroscience, vol. 21, no. 6, pp. 860-868, 2018.

[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," arXiv preprint arXiv:1803.11347, 2018.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks." pp. 1126-1135.

[7] J. Ho, and S. Ermon, "Generative adversarial imitation learning." pp. 4565-4573.

[8] H. Li, X. Liao, and L. Carin, "Multi-task reinforcement learning in partially observable stochastic environments," Journal of Machine Learning Research, vol. 10, no. May, pp. 1131-1186, 2009.

[9] F. Fernández, and M. Veloso, "Learning domain structure through probabilistic policy reuse in reinforcement learning," Progress in Artificial Intelligence, vol. 2, no. 1, pp. 13-27, 2013.

[10] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, "Multi-task policy search for robotics." pp. 3876-3881.

[11] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," arXiv preprint arXiv:1511.06342, 2015.

[12] J. Schmidhuber, "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook," Technische Universität München, 1987.

[13] S. Thrun, and L. Pratt, Learning to learn: Springer Science & Business Media, 2012.

[14] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent." pp. 87-94.

[15] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent." pp. 3981-3989.

[16] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, "Matching networks for one shot learning." pp. 3630-3638.

[17] S. Ravi, and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[18] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks." pp. 1842-1850.

[19] I. Clavera, A. Nagabandi, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt: Meta-learning for model-based control," arXiv preprint arXiv:1803.11347, vol. 3, 2018.

[20] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network." pp. 305-313.

[21] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning." pp. 627-635.

[22] P. Abbeel, and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning." p. 1.

[23] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." pp. 1433-1438.

[24] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes." pp. 19-27.

[25] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization." pp. 49-58.

[26] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning." pp. 729-736.

[27] E. Klein, M. Geist, B. Piot, and O. Pietquin, "Inverse reinforcement learning through structured classification." pp. 1007-1015.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets." pp. 2672-2680.

[29] D. P. Kingma, and M. Welling, "Auto-encoding variational bayes," arXiv preprint arXiv:1312.6114, 2013.

[30] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," arXiv preprint arXiv:1802.01557, 2018.

[31] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," arXiv preprint arXiv:1709.04905, 2017.

[32] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning." pp. 1087-1098.

[33] A. Gleave, and O. Habryka, "Multi-task maximum entropy inverse reinforcement learning," arXiv preprint arXiv:1805.08882, 2018.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.

[35] S. Hochreiter, and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[36] Y. Li, H. Wang, B. Ding, and W. Zhou, "RoboCloud: augmenting robotic visions for open environment modeling using Internet knowledge," Science China Information Sciences, vol. 61, no. 5, pp. 050102, 2018.

[37] X. Gong, B. Ding, J. Xu, H. Wang, X. Zhou, and D. Feng, "Parallelized Synchronous Multi-agent Deep Reinforcement Learning with Experience Replay Memory." pp. 325-3255.