Meta Exploration for Model-Agnostic Reinforcement Learning

Swaminathan Gurumurthy (sgurumur) [1] **Sumit Kumar (skumar2)** [1] **Tianwei Yue (tyue)** [1]

Abstract

Meta-Reinforcement learning approaches aim to develop learning procedures that can adapt quickly to a distribution of tasks with the help of a few examples. Developing efficient exploration strategies capable of finding the most useful samples becomes critical in such settings. Existing approaches towards finding efficient exploration strategies add auxiliary objectives to promote exploration by the pre-update policy, however, this makes the adaptation using a few gradient steps difficult as the pre-update (exploration) and post-update (exploitation) policies are quite different. Instead of sticking to methods for more sufficient policy adaption, we propose to explicitly model a separate exploration policy with task-specific variables z for the task distribution. Having two different policies gives more flexibility in training the exploration policy and also makes adaptation to any specific task easier. We also use DiCE operator to ensure that the gradients of z can be properly back-propagated. We show that using self-supervised or supervised learning objectives for adaptation stabilizes the training process and also demonstrate the superior performance of our model compared to prior works in this domain.

1. Introduction

Reinforcement learning (RL) approaches have seen many successes in recent years, from mastering the complex game of Go (Silver et al., 2017) to even discovering molecules (Olivecrona et al., 2017). However, a common limitation of these methods is their propensity to overfitting on a single task and inability to adapt to even slightly perturbed configuration (Zhang et al., 2018). On the other hand, humans have this astonishing ability to learn new tasks in a matter of minutes by using their prior knowledge and understanding of the underlying task mechanics. Drawing inspiration from human behaviors, researchers have proposed to incorporate multiple inductive biases and heuristics to help the models learn quickly and generalize to unseen scenarios. Despite a lot of effort it has been difficult to approach human levels of data efficiency and generalization.

Meta reinforcement learning addresses these shortcomings by learning how to learn these inductive biases and heuristics from the data itself. It strives to learn an algorithm that allows an agent to succeed in a previously unseen task or environment when only limited experience is available. These inductive biases or heuristics can be induced in the model in various ways like optimization algorithm, policy, hyperparameters, network architecture, loss function, exploration strategies *etc*. Recently, a class of initialization based meta-learning approaches have gained attention like Model Agnostic Meta-Learning (MAML) (Finn et al., 2017). MAML finds a good initialization for a model or a policy that can be adapted to a new task by fine-tuning with policy gradient updates from a few samples of that task.

Since the objective of meta-RL algorithms is to adapt to a new task from a few examples, efficient exploration strategies are crucial for quickly finding the optimal policy in a new environment. Some recent works (Gupta et al., 2018a;b; Rakelly et al., 2019) have tried to address this problem by using latent variables to model the distribution of exploration behaviors. Another set of approaches (Stadie et al., 2018; Rothfuss et al., 2018) focus on improving the credit assignment of the meta learning objective to the pre-update trajectory distribution.

However, all these prior works on gradient based Meta-RL use one or few policy gradient updates to transition from pre- to post-update policy. This limits the applicability of these methods to cases where the post-update (exploitation) policy is similar to the pre-update (exploration) policy and can be obtained with only a few updates. Also, for cases where pre- and post-update policies are expected to exhibit different behaviors, large gradient updates may result in training instabilities and lack of convergence.

To address this problem, we propose to explicitly model a separate exploration policy for the distribution of tasks. The exploration policy is trained to find trajectories that can lead to fast adaptation of the exploitation policy on the given task. This formulation provides much more flexibil-

[1]Carnegie Mellon University, Pittsburgh, PA 15213, USA.

10708 Class Project, Spring, 2019.

ity in training the exploration policy, which we will show to improve sample efficiency. We will further show that, in order to adapt as quickly as possible to the new task, it is often more useful to use self-supervised or supervised learning approaches, where possible, to lead to more stable gradient update steps while training exploitation policy with the trajectories collected from a different exploration policy.

2. Related work

Meta-Learning. Meta-learning aims to develop learning procedures flexible under the given domain or task (Vilalta & Drissi, 2002), and it tries to develop learning procedures for fast adaptation to new problem or unseen data. Though learning to perform proper and universal tuning from the trained parameters is capable of efficient model adaptation (Hochreiter et al., 2001; Andrychowicz et al., 2016), the method is far from being effective in RL problems where models need to be trained for optimization over long horizons or trajectories.

Meta-Reinforcement Learning. Meta-learning algorithms proposed in the RL community include approaches based on recurrent models (Duan et al., 2016b; Finn & Levine, 2017), metric learning (Snell et al., 2017; Sung et al., 2018), and learning optimizers (Nichol et al., 2018). To achieve quick adaptation, efficient exploration strategies are crucial for quickly finding the optimal policy in a new environment. While previous heuristic exploration methods in standard RL settings include max-entropy RL (Haarnoja et al., 2017; 2018; Eysenbach et al., 2018), curiosity-driven methods (Pathak et al., 2017), *etc.*, some recent methods towards efficient exploration under meta-RL context include designing synthetic rewards (Xu et al., 2018), model-based RL with latent Gaussian processes variable (Sæmundsson et al., 2018), modular (Alet et al., 2018) or hierarchical policy search (Nachum et al., 2018), context-based graphical models with task-specific latent representations (Rakelly et al., 2019).

Gradient-Based Meta-RL. A set of recently proposed meta-gradient-based approaches following Model Agnostic Meta-Learning (MAML) (Finn et al., 2017) have shown successes in RL problems. MAML aims to find a good initialization for a policy or a model that can be adapted to a new task by fine-tuning with policy gradient updates from a few samples of that task. Built on top of MAML (Finn et al., 2017) , Gupta et al. (2018b) proposed structured exploration strategies with latent variables. Gupta et al. (2018a) performed an unsupervised policy learning from the broad task distribution acquired via diversity-driven exploration (Eysenbach et al., 2018).

However, many of the previously mentioned gradient-based meta learning methods involves poor credit assignment to pre-adaptation behaviors, which results in poor sample efficiency as well as ineffective task identification. Wortsman et al. (2018) first followed the MAML objective to let the agent interact with the environment to obtain the adapted parameters, then they learned to what extent should they adapt by minimizing the navigation loss and a self-supervised interaction loss. E-MAML (Stadie et al., 2018) considered per-task sampling distributions as extra information for exploration. They explicitly optimizes the per-task sampling distributions during adaptation with respect to the expected future returns produced by the post-adaptation policy. Rothfuss et al. (2018) proposed proximal meta-policy search (ProMP) to sufficiently and properly react to pre-update sampling distribution in order to decide the extent of policy adaptation in each step. They also designed low variance curvature surrogate objective to control the variance of policy gradient update, and they promoted the meta-learning stability by controlling the divergence of both pre and post-update policies.

Building on the work of ProMP (Rothfuss et al., 2018), we want to explore the chances of using separate exploration policy instead of pre-update policy to sample trajectories for gradient update, and further explore the approaches towards more stable meta gradient update steps for exploitation policy adaptation from trajectories sampled by exploration policy.

3. Background

3.1. Reinforcement Learning

In the context of a discrete-time finite Markov decision process (MDP), consider a task $\mathcal{T}$ defined by the tuple $(\mathcal{S}, \mathcal{A}, \boldsymbol{P}, r, \gamma, H)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $\boldsymbol{P}$, reward function r, discount factor γ, and time horizon H. A trajectory $\boldsymbol{\tau} := (s_0, a_0, r_0, \ldots, s_{H-1}, a_{H-1}, r_{H-1}, s_H)$ is a sequence of state s_t, action a_t and reward r_t received for the entire time horizon H. The task return under the trajectory $\boldsymbol{\tau}$ is defined as the sum of discounted rewards encountered along that trajectory: $R(\boldsymbol{\tau}) = \sum_{t=1}^{H} \gamma^t r_t$. The goal of reinforcement learning is to find the optimal policy $\pi(a|s)$ that maximizes the expected task return $\mathbb{E}_{\tau \sim P_{\mathcal{T}}(\tau|\pi)}[R(\boldsymbol{\tau})]$.

3.2. Meta-Reinforcement Learning

Unlike RL which tries to find an optimal policy for a single task, meta-RL aims to find a policy that can generalize to a distribution of tasks. Each task $\mathcal{T}$ sampled from the distribution $\rho(\mathcal{T})$ corresponds to a different Markov Decision Process (MDP). These MDPs have similar state and action space but might differ in the reward function r or the environment dynamics $\boldsymbol{P}$. The goal of meta RL is to quickly adapt the policy to any task $\mathcal{T} \sim \rho(\mathcal{T})$ with the help of few examples from that task.

3.3. Credit Assignment in Meta-RL

Finn et al. (2017) introduced MAML - a gradient-based meta-RL algorithm that tries to find a good initialization for a policy which can be adapted to any task $\mathcal{T} \sim \rho(\mathcal{T})$ by fine-tuning with one or more gradient updates using the sampled trajectories of that task. MAML maximizes the following objective function:

$$J(\theta) = \mathbb{E}_{\mathcal{T}\sim\rho(\mathcal{T})}\left[\mathbb{E}_{\boldsymbol{\tau}'\sim P_{\mathcal{T}}(\boldsymbol{\tau}'|\theta')}\left[R(\boldsymbol{\tau}')\right]\right]$$
$$\text{with} \quad \theta' := U(\theta,\mathcal{T}) = \theta + \alpha\nabla_\theta \mathbb{E}_{\boldsymbol{\tau}\sim P_{\mathcal{T}}(\boldsymbol{\tau}|\theta)}\left[R(\boldsymbol{\tau})\right] \quad (1)$$

where U is the update function that performs one policy gradient ascent step to maximize the expected reward $R(\boldsymbol{\tau})$ obtained on the trajectories $\boldsymbol{\tau}$ sampled from task $\mathcal{T}$. Essentially, MAML tries to find a good initialization θ for a policy which is fine-tuned to θ' by performing the policy gradient update $U(\theta,\mathcal{T})$ on the trajectories $\boldsymbol{\tau}$ sampled from the task $\mathcal{T}$.

Rothfuss et al. (2018) showed that the gradient of the objective function $J(\theta)$ in eq. 1 can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathcal{T}\sim\rho(\mathcal{T})}\left[\mathbb{E}_{\substack{\boldsymbol{\tau}\sim P_{\mathcal{T}}(\boldsymbol{\tau}|\theta)\\ \boldsymbol{\tau}'\sim P_{\mathcal{T}}(\boldsymbol{\tau}'|\theta')}}\left[\nabla_\theta J_{\text{post}}(\boldsymbol{\tau},\boldsymbol{\tau}') + \nabla_\theta J_{\text{pre}}(\boldsymbol{\tau},\boldsymbol{\tau}')\right]\right]$$

where,

$$\nabla_\theta J_{\text{post}}(\boldsymbol{\tau},\boldsymbol{\tau}') = \underbrace{\nabla_{\theta'}\log\pi_\theta(\boldsymbol{\tau}')R(\boldsymbol{\tau}')}_{\nabla_{\theta'}J^{\text{outer}}} \underbrace{\left(I + \alpha R(\boldsymbol{\tau})\nabla_\theta^2\log\pi_{\theta'}(\boldsymbol{\tau})\right)}_{\text{transformation from }\theta'\text{ to }\theta}$$

$$\nabla_\theta J_{\text{pre}}(\boldsymbol{\tau},\boldsymbol{\tau}') = \alpha\nabla_\theta\log\pi_\theta(\boldsymbol{\tau})\left(\underbrace{(\nabla_\theta\log\pi_\theta(\boldsymbol{\tau})R(\boldsymbol{\tau}))^\top}_{\nabla_\theta J^{\text{inner}}} \underbrace{(\nabla_{\theta'}\log\pi_{\theta'}(\boldsymbol{\tau}')R(\boldsymbol{\tau}'))}_{\nabla_{\theta'}J^{\text{outer}}}\right)$$

The first term $\nabla_\theta J_{\text{post}}(\boldsymbol{\tau},\boldsymbol{\tau}')$ corresponds to a policy gradient step on the post-update policy $\pi_{\theta'}$ with respect to the post-update parameters θ' which is then followed by a linear transformation from θ' to θ (pre-update parameters). Note that $\nabla_\theta J_{\text{post}}(\boldsymbol{\tau},\boldsymbol{\tau}')$ optimizes θ to increase the likelihood of the trajectories $\boldsymbol{\tau}'$ that lead to higher returns given some trajectories $\boldsymbol{\tau}$. However, this term does not optimize θ to yield trajectories $\boldsymbol{\tau}$ that lead to good adaptation steps. That is infact, done by the second term $\nabla_\theta J_{\text{pre}}(\boldsymbol{\tau},\boldsymbol{\tau}')$. It optimizes for the pre-update trajectory distribution, $P_{\mathcal{T}}(\boldsymbol{\tau}|\theta)$, i.e, increases the likelihood of trajectories $\boldsymbol{\tau}$ that lead to good adaptation steps.

During optimization, MAML only considers $J_{\text{post}}(\boldsymbol{\tau},\boldsymbol{\tau}')$ and ignores $J_{\text{pre}}(\boldsymbol{\tau},\boldsymbol{\tau}')$. Thus MAML finds a policy that adapts quickly to a task given relevant experiences, however, the policy is not optimized to gather useful experiences from the environment that can lead to fast adaptation.

Rothfuss et al. (2018) proposed Proximal Meta Policy Search (ProMP) where they analyze this issue with MAML and incorporates $\nabla_\theta J_{\text{pre}}(\boldsymbol{\tau},\boldsymbol{\tau}')$ term in the update as well. They used The Infinitely Differentiable Monte-Carlo Estimator (DICE) (Foerster et al., 2018) to allow causal credit assignment on the pre-update trajectory distribution, however, the gradients computed by DICE suffer from high variance estimates. To remedy this, they proposed a low variance (and slightly biased) approximation of the DICE based loss that leads to stable updates.

However, the pre-update and post-update policies are often expected to exhibit very different behaviors, i.e, exploration and exploitation behaviors respectively. For instance, consider a 2D environment where a task corresponds to reaching a goal location sampled randomly from the four corner regions. The agent receives a reward only if it lies in some vicinity of the goal location. The optimal pre-update or exploration policy will be to visit each of the four corner regions sequentially whereas the ideal post-update or exploitation policy will be to reach the goal state as fast as possible once the goal region is discovered. Clearly, the two policies are expected to behave very differently. In such cases, transitioning a single policy from pure exploration phase to pure exploitation phase via policy gradient updates will require multiple steps. Unfortunately, this significantly increases the computational and memory complexities of the algorithm. Furthermore, it may not even be possible to achieve this transition via few gradient updates. This raises an important question: **Do we really need to use the pre-update policy for exploration as well? Can we use a separate policy for exploration?**

Another problem with these approaches stems from the high variance nature of the inner loop policy gradient updates resulting in both $\nabla_\theta J_{\text{pre}}$ and $\nabla_\theta J_{\text{post}}$ terms being high variance. Note that $\nabla_\theta J_{\text{pre}}$ computes the inner product between pre-update and post-update policy gradients. Since, the two policy gradient terms have been computed using two different policies (i.e. pre-update and post-update policies) on two different trajectory distributions, the resulting updates are highly likely to be unstable. ProMP tries to stabilize the updates by adding trust region constraints, however, that doesn't completely solve the issue. This motivates another question: **Do we really need the pre-update**

gradients to be policy gradients? Can we use a different objective in the inner loop to get more stable updates?

4. Method

In this section, we describe our proposed model that address the two queries we raised in the previous section.

Using separate policies for pre-update and post-update sampling: The straightforward solution to the first problem mentioned above is to use a separate exploration policy μ_ϕ responsible for collecting trajectories for the inner loop updates. Following that, post-update policy π_θ can be used to collect trajectories for performing the outer loop updates. Unfortunately, this is not as simple as it sounds. To understand this, let's look at the inner loop updates:

$$U(\theta, \mathcal{T}) = \theta + \alpha \nabla_\theta \mathbb{E}_{\boldsymbol{\tau} \sim P_{\mathcal{T}}(\boldsymbol{\tau}|\theta)} \left[R(\boldsymbol{\tau}) \right]$$

When the exploration policy is used for sampling trajectories, we need to perform importance sampling. The update would thus become:

$$U(\theta, \mathcal{T}) = \theta + \alpha \nabla_\theta \mathbb{E}_{\boldsymbol{\tau} \sim Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)} \left[\frac{P_{\mathcal{T}}(\boldsymbol{\tau}|\theta)}{Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)} R(\boldsymbol{\tau}) \right]$$

where $P_{\mathcal{T}}(\boldsymbol{\tau}|\theta)$ and $Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)$ represent the trajectory distribution sampled by π_θ and μ_ϕ respectively. Note that the above update is an off-policy update which results in high variance estimates when the two trajectory distributions are quite different from each other. This makes it infeasible to use the imporatance sampling update in the current form. To address this instability issue, let us consider the second question we raised before.

Using a self-supervised/supervised objective for the inner loop update step: The instability in the inner loop updates arises due to the high variance nature of the policy gradient update. Note that the objective of inner loop update is to provide some task specific information to the agent with the help of which it can adapt its behavior in the new environment. One potential solution of the high variance update is to use some form of self-supervised or supervised learning objective in place of policy gradient in the inner loop to ensure that the updates are more stable. We propose to use a network for predicting some task (or MDP) specific property like reward function, expected return or value. During the inner loop update, the network updates its parameters by minimizing its prediction error on the given task. Unlike prior meta-RL works where the task adaptation in the inner loop is done by policy gradient updates, here, we perform gradient descent on a supervised loss objective function resulting in stability during the adaptation phase.

4.1. Model

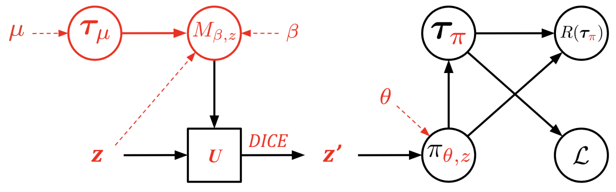


Figure 1. Model Flowchart: Black structures are those consistent with E-MAML/ProMP. Red structures are those differs from E-MAML/ProMP. The thin-dotted arrow means the parameters related to that node.

Our proposed model comprises of three modules, the exploration policy $\mu_\phi(s)$, the exploitation policy $\pi_{\theta,z}(s)$, and the self-supervision network $M_{\beta,z}(s,a)$. Note that $M_{\beta,z}$ and $\pi_{\theta,z}$ share a set of parameters z while containing their own set of parameters β and θ respectively. Our model differs from E-MAML/ProMP mainly because of the separate exploration policy and the task-specific latent variable z, and the detailed design comparisons are shown in Fig. 1.

The agent first collects a set of trajectories $\boldsymbol{\tau}$ using its exploration policy μ_ϕ for each task $\mathcal{T} \sim \rho(\mathcal{T})$. It then updates the shared parameter z by minimizing the regression loss $(M_{\beta,z}(s,a) - M^t(s,a))^2$ as shown:

$$z' = U(z, \mathcal{T}) = z - \alpha \nabla_z \mathbb{E}_{\boldsymbol{\tau} \sim Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)} \left[\sum_{t=0}^{H-1} \left(M_{\beta,z}(s_t,a_t) - \overline{M}(s_t,a_t) \right)^2 \right]$$

where, $\overline{M}(s,a)$ is the target which can be any of the task specific quantity like reward, return, value, etc. We further modify the above equation by multiplying the DICE operator to simplify the gradient computation with respect to ϕ as DICE allows it to be computed in a straightforward manner with back-propagation. This also eliminates the need to apply the policy gradient trick to expand the above expression for gradient computation. The update then becomes:

$$z' = U(z, \mathcal{T}) = z - \alpha \nabla_z \mathbb{E}_{\boldsymbol{\tau} \sim Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)} \left[\sum_{t=0}^{H-1} \left(\prod_{t'=0}^{t} \frac{\mu_\phi(a_{t'}|s_{t'})}{\perp(\mu_\phi(a_{t'}|s_{t'}))} \right) \left(M_{\beta,z}(s_t,a_t) - \overline{M}(s_t,a_t) \right)^2 \right]$$

where $\perp$ is the stop gradient operator as introduced in Foerster et al. (2018). After obtaining the updated parameters z', the agent samples trajectories $\boldsymbol{\tau}'$ using its updated exploitation policy $\pi_{\theta,z'}$. Note that our model enables the agent to learn a generic exploitation policy $\pi_{\theta,z}$ for the task distribution which can then be adapted to any specific task by updating z to z' as shown above. Effectively, z' encodes the necessary information regarding the task that helps an

Algorithm 1 Model-Agnostic Meta-Exploration

```
1: Require: Task distribution $\rho(\mathcal{T})$, step sizes $\alpha$, $\eta$
2: while not converge do
3:     Sample a batch of tasks $\mathcal{T}_i \sim \rho(\mathcal{T})$
4:     for all sampled tasks $\mathcal{T}_i$ do
5:         Collect pre-update trajectories $\boldsymbol{\tau}_{\mu}^{\mathcal{T}_i}$ using $\mu_\phi(s)$
6:         Update $z$ by minimizing $\left(M_{\beta,z}(s,a) - M^t(s,a)\right)^2$:
               $$z' = U(z,\mathcal{T}) = z - \alpha\nabla_z \mathbb{E}_{\boldsymbol{\tau}_{\mu}^{\mathcal{T}_i} \sim Q_{\mathcal{T}_i}(\boldsymbol{\tau}|\phi)}\left[\sum_{t=0}^{H-1}\left(\prod_{t'=0}^{t}\frac{\mu_\phi(a_{t'}|s_{t'})}{\perp(\mu_\phi(a_{t'}|s_{t'}))}\right)\left(M_{\beta,z}(s,a) - M^t(s,a)\right)^2\right]$$
7:         where $\left(\prod_{t'=0}^{t}\frac{\mu_\phi(a_{t'}|s_{t'})}{\perp(\mu_\phi(a_{t'}|s_{t'}))}\right)$ is the DiCE operator
8:         Collect post-update trajectory $\boldsymbol{\tau}_{\pi}^{\mathcal{T}_i}$ using $\pi_{\theta,z'}(s)$
9:     Policy gradient update w.r.t post-update trajectory $\boldsymbol{\tau}_{\pi}^{\mathcal{T}_i}$ to optimize all parameters $z, \theta, \phi, \beta$, with the objective
               $$J(z',\theta) = \mathbb{E}_{\mathcal{T}\sim\rho(\mathcal{T})}\left[\mathbb{E}_{\boldsymbol{\tau}_{\pi}^{\mathcal{T}} \sim P_{\mathcal{T}}(\boldsymbol{\tau}_{\pi}^{\mathcal{T}}|\theta,z')}\left[R(\boldsymbol{\tau}_{\pi}^{\mathcal{T}})\right]\right]$$
```

agent in adapting its behavior to maximize its expected return.

After obtaining the updated parameters z', the agent uses the exploitation policy $\pi_{\theta,z'}$, parameterized by the updated parameters z' to collect the validation trajectories $\boldsymbol{\tau}_{\pi}^{\mathcal{T}}$ for each task $\mathcal{T}$. The collected trajectories are then used to perform a policy gradient update to all parameters z, θ, ϕ and β using the gradients computed by back-propagation for the following policy gradient objective:

$$J(z',\theta) = \mathbb{E}_{\mathcal{T}\sim\rho(\mathcal{T})}\left[\mathbb{E}_{\boldsymbol{\tau}_{\pi}^{\mathcal{T}} \sim P_{\mathcal{T}}(\boldsymbol{\tau}_{\pi}^{\mathcal{T}}|\theta,z')}\left[R(\boldsymbol{\tau}_{\pi}^{\mathcal{T}})\right]\right]$$

Unlike traditional meta-RL algorithms where both the inner-loop and outer-loop updates are policy gradient updates, in this work, the inner-loop update is a supervised learning gradient descent update whereas the outer-loop update remains to be a policy gradient update. The pseudo-code of our algorithm is shown in algorithm 1.

We found that implementing algorithm 1 as it is leads to high variance DICE gradients. This consequently leads to high variance gradients for the ϕ as well resulting in instability during training and poor performance of the learned model. To alleviate this, we apply some variance reduction techniques as described below.

The vanilla DICE gradients can be written as follows:

$$\nabla_\phi J(z',\theta) = \mathbb{E}_{\mathcal{T}\sim\rho(\mathcal{T})}\Bigg[\mathbb{E}_{\boldsymbol{\tau}\sim Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)}\sum_{t=0}^{H-1}\alpha\nabla_\phi\log\mu_\phi(s_t) \left[\sum_{t'=t}^{H-1}\left(\mathbb{E}_{\boldsymbol{\tau}'\sim P_{\mathcal{T}}(\boldsymbol{\tau}'|\theta,z')}\left(\nabla_{z'}\log\pi_{\theta,z'}(\boldsymbol{\tau}')R(\boldsymbol{\tau}')\right)^\top\right) \left(\nabla_z\left(M_{\beta,z}(s_t,a_t) - \overline{M}(s_t,a_t)\right)^2\right)\right]\Bigg]$$

The above expression can be viewed as a policy gradient update:

$$\nabla_\phi J(z',\theta) = \mathbb{E}_{\mathcal{T}\sim\rho(\mathcal{T})} \left[\mathbb{E}_{\boldsymbol{\tau}\sim Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)}\sum_{t=0}^{H-1}\alpha\nabla_\phi\log\mu_\phi(s_t)R_t^\mu\right] \quad (2)$$

with returns

$$R_t^\mu = \left[\sum_{t'=t}^{H-1}\left(\mathbb{E}_{\boldsymbol{\tau}'\sim P_{\mathcal{T}}(\boldsymbol{\tau}'|\theta,z')}\left(\nabla_{z'}\log\pi_{\theta,z'}(\boldsymbol{\tau}')R(\boldsymbol{\tau}')\right)^\top\right) \left(\nabla_z\left(M_{\beta,z}(s_t,a_t) - \overline{M}(s_t,a_t)\right)^2\right)\right] \quad (3)$$

Drawing inspiration from Mnih et al. (2016), we replace the return R_t^μ in 3 with an advantage estimate A_t^μ and apply the following gradients:

$$\nabla_\phi \hat{J}(z',\theta) = \mathbb{E}_{\mathcal{T}\sim\rho(\mathcal{T})}\left[\mathbb{E}_{\boldsymbol{\tau}\sim Q_{\mathcal{T}}(\boldsymbol{\tau}|\phi)}\sum_{t=0}^{H-1}\alpha\nabla_\phi\log\mu_\phi(s_t)A_t^\mu\right]$$

where,

$$A_t^\mu = r_t^\mu + V_{t+1}^\mu - V_t^\mu \quad (4)$$

where V_t^μ is computed using using a linear feature baseline (Duan et al., 2016a) fitted on the returns R_t^μ and r_t^μ is given by:

$$r_t^\mu = \Big(\mathbb{E}_{\boldsymbol{\tau}' \sim P_{\mathcal{T}}(\boldsymbol{\tau}'|\theta,z')}\left(\nabla_{z'} \log \pi_{\theta,z'}(\boldsymbol{\tau}') R(\boldsymbol{\tau}')\right)^\top \Big) \Big(\nabla_z \left(M_{\beta,z}(s_t,a_t) - \overline{M}(s_t,a_t)\right)^2\Big)\Big) \quad (5)$$

5. Experiments

We have evaluated our proposed model on the environments used by Finn et al. (2017) and Rothfuss et al. (2018). Specifically, we use the following environments:

- *Point navigation task* - Each task corresponds to reaching a randomly sampled goal location in a 2D environment. The goal location is present in one of the four corner regions of the environment. This is a sparse reward task where the agent receives a reward only if it is sufficiently close to the goal location. In other words, it is the correct corner region that contains the goal. Thus, to successfully identify the task, the agent must learn to efficiently explore the different corner regions in the sparse reward task and then proceed to the goal.
- *Meta-RL benchmark continuous control tasks* - We tested our proposed model on 3 Continuous control Mujoco environments : AntRandDir, HalfCheetahVel and HalfCheetahFwdBack. All of these environments are dense reward ones.
  - AntRandDir - A 6 legged agent is specified one of the 2 directions (forward or backward) and is rewarded if it goes in that direction.
  - HalfCheetahVel - A 2 legged agent is specified a particular velocity in range (0,3). It is rewarded if it goes at that velocity.
  - HalfCheetahFwdBack - A 2 legged agent is specified one of the 2 directions (forward or backward) and is rewarded if it goes in that direction.

We noticed that the dense reward environments used in Rothfuss et al. (2018) and Finn et al. (2017) had different reward functions. We chose to use the environments used in Finn et al. (2017) for our experiments. That might be the reason for the instability in ProMP. However that also speaks to the immense sensitivity of their models to hyperparameters. In fact, even the experiments we ran on their own environments showed similar unstable behavior.

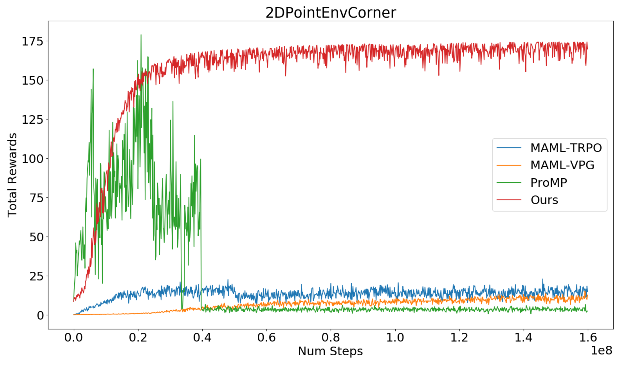
We treat the shared parameter z as a learnable latent embedding. The exploitation policy $\pi_{\theta,z}(s)$ and the self-supervision network $M_{\beta,z}(s,a)$ concatenates z with their respective inputs. All the three networks have the same architecture as that of the policy network in Finn et al. (2017). For the self-supervision network, we use the returns calculated at each time step of the exploration policy as the self-supervision signal for the adaptation phase.

We compare our model against MAML-TRPO, MAML-VPG (vanilla policy gradient) and ProMP. Note that we used the same hyperparameters as specified in the MAML repository[1]. We used the offical ProMP repository[2] for running their method on all the environments. Also, we restrict ourselves to a single adaptation step in all environments, but it can be easily extended to multiple gradient steps as well. For our model as well, we restricted to the same hyperparameters used by [3], but instead of using conjugate gradient updates we used Adam (Kingma & Ba, 2014) optimizer with a learning rate of $7e-4$ for all parameters except z which uses $4e-5$.

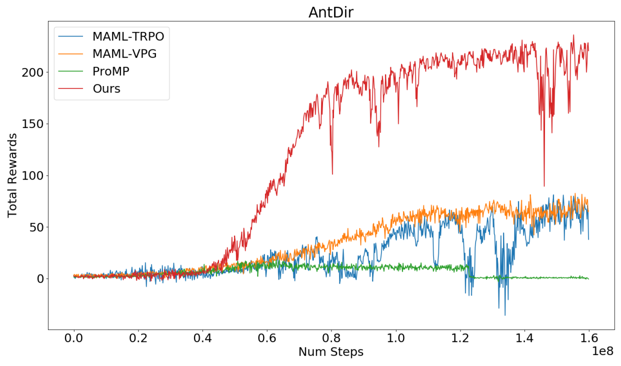
5.1. Dense Reward Tasks

Meta RL Benchmark Continuous Control Tasks. We show the results for the Dense reward environments from the Meta RL benchmark tasks, namely HalfCheetahDir, HalfCheetahVel and AntDir. The performance plots for all the 4 algorithms are shown in Fig. 2. In all the environments, our proposed method outperforms others in terms of asymptotic performance and sample complexity. As evident from the figures, the training is much more stable for our method especially when compared with ProMP. This can also be noted from the much lower variance of our updates compared to the baselines. Although ProMP manages to reach similar peak performance to our method in HalfCheetahFwdBack and HalfCheetahVel, the training itself is pretty unstable indicating the inherent fragility of their updates. We noticed that the performance of ProMP decreases suddenly during the training in the two Half Cheetah environments. This might be due to the fact that the results reported in the ProMP work are on environments whose rewards have been modified from the environments used in MAML, but we used the same hyperparameters as reported in their official repository. With significant hyperparameter tuning, one can expect to see stable training of ProMP on these environments as well, however, that in itself speaks about the sensitivity of the algorithm with respect to hyperparameters. Our algorithm particularly performs well in AntDir environment where the rewards are largely uninformative in the initial stages of training given

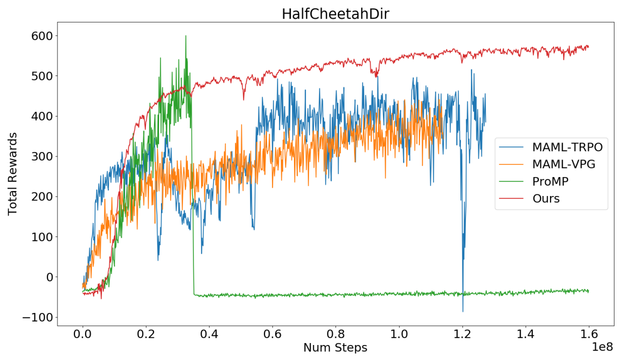
[1] https://github.com/cbfinn/maml_rl
[2] https://github.com/jonasrothfuss/ProMP
[3] https://github.com/cbfinn/maml_rl



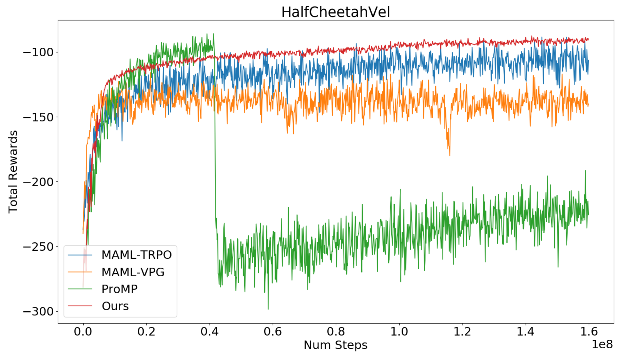
(a) 2D Point Navigation



(b) AntDirEnv



(c) HalfCheetahDirEnv



(d) HalfCheetahVelEnv

Figure 2. Comparison of our method with 3 baseline methods in sparse and dense reward environments.

the large state space. Thus, here as well, exploration and stable updates become critical thus allowing our algorithm to perform much better than the baselines.

5.2. Sparse Reward Tasks

2D Point Navigation. We show the performance plots for all four algorithms in the sparse reward 2DPointEnvCorner in Fig. 2. The difference between our method and the baselines is even more significant in this environment given the sparse reward scenario. In this environment, the agent needs to perform efficient exploration and use the sparse reward trajectories to perform stable updates both of which are salient aspects of our algorithm. Our method is able to achieve this as it evident from the plots. It is able to reach peak performance and show stable behavior. ProMP, on the other hand, shows much more unstable behavior than in the dense reward scenarios, although it manages to reach similar peak performance to our method. The other baselines struggle to do much in the environment since they do not explicitly incentivize exploration for the pre-update policy.

5.3. Ablation Study

We perform several ablation experiments to analyze the impact of different components of our algorithm on 2D point navigation task. Fig. 3 shows the performance plots for the following 5 different variants of our algorithm:

VPG-Inner loop : The semi-supervised/supervised loss in the inner loop is replaced with the vanilla policy gradient loss as in MAML while using the exploration policy to sample the pre-update trajectories. This variant illustrates our claim of unstable inner loop updates when naively using an exploration policy. As expected, this model performs poorly due to the high variance off-policy updates in the inner loop.

Reward Self-Supervision : A reward based self-supervised objective is used instead of return based self-supervision, i.e, the self-supervision network M now predicts the reward instead of the return at each time step. This variant performs reasonably well but struggles to reach peak performance since the task is sparse reward. This shows that the choice of self-supervision objective is also important and needs to be chosen carefully.

Vanilla DiCE : In this variant, we directly use the DICE

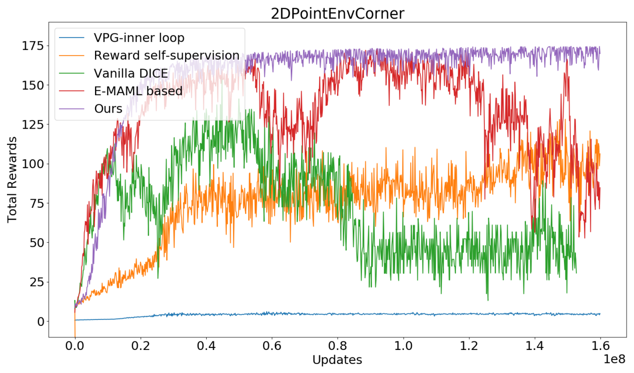


Figure 3. Ablation results

gradients to perform updates on ϕ instead of using the low variance gradient estimator. The leads to higher variance updates and unstable training as can be seen from the plots. This shows that the low variance gradient estimate had a major contribution to the stability during training.

E-MAML Based : Used an E-MAML (Stadie et al., 2018) type objective to compute the gradients w.r.t ϕ instead of using DICE, i.e, directly used policy gradient updates on μ_ϕ but instead with returns computed on post-update trajectories. This variant ignores the causal credit assignment from output to inputs. Thus, the updates are of higher variance, leading to more unstable updates, although it manages to reach peak performance.

Ours : Used the low variance estimate of the DICE gradients to compute updates for ϕ along with return based self-supervision for inner loop updates. Our model reaches peak performance and exhibits stable training due to low variance updates.

These ablations illustrate that the self-supervised objective had a huge role to play in improving stability in the updates. In fact, even the choice of the self-supervised objective can be critical in some cases (e.g, predicting reward v/s return). Further, we also show that the updates on exploration policy are also critical. The experiments above show that the variance reduction techniques used in the objective of exploration policy also have a huge impact on the performance.

6. Discussions

The above experiments illustrate that our approach provides significantly more stable updates as compared to ProMP and allows for much faster training the the MAML variants especially when the rewards aren't informative enough or sparse. More importantly, this (although not concretely) shows that in most of these tasks the exploration and exploitation policies were indeed significantly different which is what led to the instability in ProMP updates. In fact, this is one of the reasons we were suspicious of the environments in the official repository of ProMP and chose not to use them. They had shaped the reward a little in AntDir and HalfCheetahDir and we suspect that this might have been done to ensure their model shows reasonable results in these environments.

Also, we would like to note that the idea of using a separate exploration and exploitation policy is much more general and doesn't need to be restricted to MAML. It is indeed surprising that the idea of exploration hasn't been explored much in other meta-learning approaches, leave alone using a separate exploration policy. Given the requirements of sample efficiency of the adaptation steps in the meta-learning setting, exploration is a very crucial ingredient and has been vastly under explored.

7. Conclusion and Future Work

Unlike conventional meta-RL approaches, we proposed to explicitly model a separate exploration policy for the task distribution. Having two different policies gives more flexibility in training the exploration policy and also makes adaptation to any specific task easier. We showed that, through various experiments on both sparse and dense reward tasks, our model outperforms previous works while also being very stable during training. This validates that using self-supervised techniques increases the stability of these updates thus allowing us to use a separate exploration policy to collect the initial trajectories.

As for future work, we plan to do the following extension :

- We plan to experiment on more environments and add auxiliary objectives for the exploration agent. Specifically we plan to design our own sparse reward environments based on the mujoco benchmark environments to demonstrate the benefits of our method in more complicated sparse reward environments
- We also intend to incorporate the PPO (Schulman et al., 2017) inspired modifications made by ProMP (Rothfuss et al., 2018) to its objective. This would allow us to perform multiple outer-loop updates to the agents after each rollout leading to faster convergence of our model.
- But more importantly, decoupling the exploration and exploitation policies allows us to perform off-policy updates. This would tremendously improve sample efficiency. Thus, we plan to test it as a natural extention of our approach.
- Explore the use of having separate exploration and exploitation policies in other meta-learning approaches

References

Alet, F., Lozano-Pérez, T., and Kaelbling, L. P. Modular meta-learning. *arXiv preprint arXiv:1806.10166*, 2018.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016a.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl ˆ2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016b.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Finn, C. and Levine, S. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Foerster, J., Farquhar, G., Al-Shedivat, M., Rocktäschel, T., Xing, E. P., and Whiteson, S. Dice: The infinitely differentiable monte-carlo estimator. *arXiv preprint arXiv:1802.05098*, 2018.

Gupta, A., Eysenbach, B., Finn, C., and Levine, S. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018a.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pp. 5307–5316, 2018b.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1352–1361. JMLR. org, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hochreiter, S., Younger, A. S., and Conwell, P. R. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pp. 87–94. Springer, 2001.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3307–3317, 2018.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL `http://arxiv.org/abs/1803.02999`.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., and Levine, S. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Sæmundsson, S., Hofmann, K., and Deisenroth, M. P. Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*, 2018.

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

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Stadie, B. C., Yang, G., Houthooft, R., Chen, X., Duan, Y., Wu, Y., Abbeel, P., and Sutskever, I. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

Vilalta, R. and Drissi, Y. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2): 77–95, 2002.

Wortsman, M., Ehsani, K., Rastegari, M., Farhadi, A., and Mottaghi, R. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. *arXiv preprint arXiv:1812.00971*, 2018.

Xu, Z., van Hasselt, H. P., and Silver, D. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2402–2413, 2018.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.