Final Report: Comprehensive Study On Meta-Learning Algorithms

Zipei Geng
zigeng@student.ethz.ch

Chuqiao Li
chuqli@student.ethz.ch

Jiacheng Qiu
jiaqiu@student.ethz.ch

Abstract

This paper studied four high-citation meta reinforcement learning algorithms comprehensively. We first reviewed the basic notions and gave out algorithmic and heuristic understandings. Then, we conducted repetition experiments and compared the four algorithms within the same environment settings. We have found different results compared to some of the original papers and raised the questions. Finally, we proposed future improvements.

1 Introduction

The core pursue of artificial intelligence is to learn fast and flexibly which is challenging, since the agent must integrate its prior experience with a small amount of new information while avoiding overfitting to the new data. The form of prior experience and new data will largely depend on the task which makes it difficult to generalize to new situations. Meta reinforcement learning (meta-RL) [Wang et al., 2017] was then developed to mitigate this problem. To specify, meta-RL considers an underlying distribution of different but closely related tasks that differ in, for instance, the reward function or the transition probabilities from one state to another. Its objective is to enable artificial agents to efficiently learn new tasks by learning strategies for fast adaptation from prior tasks [Liu et al., 2019].

Given the above background, we seek deeper comprehension of meta-RL by reading recent publications. We will mainly review four related papers which all lie in the area of meta reinforcement learning. First, Finn et al. [2017] proposed a meta-learning algorithm that is general and *model-agnostic* which can be abbreviated as *MAML*, in the sense that it can be directly applied to any learning problem and model that is trained with a gradient descent procedure. The key idea underlying the MAML is to train the model's initial parameters such that the model has maximal performance on a new task after the parameters have been updated through one or more gradient steps computed with a small amount of data from that new task. Secondly, Mitchell et al. [2021] proposed an offline meta-RL setting and the corresponding algorithm, which is an optimization-based meta-learning algorithm that uses simple, supervised regression objectives for both the inner and outer loop of meta-training. The resulting meta-RL algorithm is abbreviated as *MACAW*. Liu et al. [2019] instead found a surrogate objective function called *Taming MAML* to replace the former MAML objective function and resulted in a more stable training via reducing the variance without introducing bias.Rakelly et al. [2019] resolved the sample efficiency problem of meta-RL by introducing an off-policy meta-RL algorithm called probabilistic embeddings for actor-critic RL (*PEARL*) that disentangles task inference and control.

In this paper, we gave out an algorithmic and intuitive understanding of the corresponding and utilized these four papers to bring the RL starters a clearer view of what is meta-RL, as well as repeat the experiments and get to know how to leverage the algorithms. The related work corresponding to the aforementioned papers is given in Section 2. The comprehension part is given in Section 3. The repetition experiments and results are given in Section 4. Finally, the discussion of future works is given in Section 5.

Foundations of Reinforcement Learning HS2021, Department of Computer Science, ETH Zürich.

2 Related Work

We mainly studied four papers and their core algorithms. Speaking of related work, we will categorize them into different related topics, instead of using that four algorithms to categorize.

Meta-learning. The core of meta-RL relies on the meta-learning framework [Schmidhuber, 1987, Thrun and Pratt, 2012] based on RL. Recently, meta-RL methods have been developed for meta-learning dynamics models [Sæmundsson et al., 2018] and policies ([Finn et al., 2017, Duan et al., 2016] that can quickly adapt to new tasks.

In meta-learning, the goal of the trained model is to quickly learn a new task from a small amount of new data, and the model is trained by the meta-learner to be able to learn on a large number of different tasks. Prior works focus on one particular learning update function or learning rule [Schmidhuber, 1987, Bengio et al., 2013, Andrychowicz et al., 2016], or placing constraints on the model architecture [Santoro et al., 2016]. In particular, gradient-based meta-RL methods learn from aggregated experience using policy gradients [Finn et al., 2017, Stadie et al., 2018, Rothfuss et al., 2018, Xu et al., 2018a], meta-learned loss functions [Sung et al., 2017], or hyperparameters [Xu et al., 2018b]. These methods focus on on-policy meta-learning. The paper of MAML gives out a meta-learning framework that is compatible with any gradient-based model and applicable to classification, regression, and RL.

Besides, recurrent [Wang et al., 2017, Duan et al., 2016] and recursive [Mishra et al., 2017] meta-RL methods are also introduced to adapt to new tasks by aggregating experience into a latent representation on which the policy is conditioned. These approaches can be categorized into context-based meta-RL methods, which is another branch of meta-RL, contrary to the gradient-based meta-RL methods. While prior work has studied methods that can train recurrent Q-functions with off-policy Q-learning methods, such methods have often been applied to much simpler tasks [Heess et al., 2015], and in discrete environments [Hausknecht and Stone, 2015].

Instead of RL, Meta-learning methods for few-shot supervised learning problems have explored a wide variety of approaches and architectures [Santoro et al., 2016, Vinyals et al., 2016, Oreshkin et al., 2020]. The permutation-invariant embedding function which is used in the paper of PEARL is inspired by the embedding function of prototypical networks [Snell et al., 2017]. While Snell et al. [2017] uses a distance metric in a learned, deterministic embedding space to classify new inputs, the embedding of PEARL is probabilistic and is used to condition the behavior of an RL agent.

Variance Reduction. As for Taming MAML, its work has related to recent work on variance reduction for policy gradient using control variates in various settings [Grathwohl et al., 2017, Wu et al., 2018]. The problem of bias-variance trade-off in policy gradient estimation has also been noticed and worked on [Heess et al., 2015]. Variance reduction for general stochastic computational graphs has also been talked about Schulman et al. [2015] and Weber et al. [2019]. Recently, Mao et al. [2019] proposes using control variates to reduce variance of high order gradient estimation.

Probabilistic meta-learning. As for the paper of PEARL, prior work has applied probabilistic models to meta-learning in both supervised and reinforcement learning domains. Hierarchical Bayesian models have been used to model few-shot learning [Tenenbaum, 1999, Fe-Fei et al., 2003], including approaches that perform gradient-based adaptation [Grant et al., 2018]. For supervised learning, Finn et al. [2018] adapt model predictions using probabilistic latent task variables inferred via amortized approximate inference. PEARL extends this idea to off-policy meta-RL. In the context of RL, Hausman et al. [2018] also conditions the policy on inferred task variables, but the aim is to compose tasks via the embedding space, while PEARL focuses on rapid adaptation to new tasks. Finally, Gupta et al. [2018] infers task variables by optimizing them with gradient descent and explores by sampling from the prior function. PEARL instead uses a different path, which leverages the posterior function.

Partially observed MDPs. Adaptation at test time in meta-RL can be viewed as a special case of RL in a POMDP [Kaelbling et al., 1998] by including the task as the unobserved part of the state. PEARL uses a variational approach to estimate belief over the task.

Offline Meta-RL. As for the paper of MACAW, prior meta-RL methods require interaction with the MDP for each of the meta-training tasks [Finn et al., 2017], and though some prior methods build on off-policy RL algorithms [Rakelly et al., 2019], these algorithms are known to perform poorly in the fully offline setting [Levine et al., 2020]. Both of the offline meta-RL settings described above inherit

the distributional difficulties of offline RL, which means that addressing this problem setting requires a new type of meta-RL method capable of meta-training on offline data.

To further address the problem of combining MAML framework with value-based RL, one might combine MAML with a supervised, bootstrap-free RL subroutine, such as advantage-weighted regression (AWR) [Peters and Schaal, 2007, Peng et al., 2019], for both for the inner and outer loop of a gradient-based meta-learning algorithm, yielding a *MAML + AWR* algorithm.

Motivated by prior work that studies the expressive power of MAML [Finn and Levine, 2017], MACAW algorithm increases the expressive power of the meta-learner by introducing a carefully chosen policy update in the inner loop.

3 Algorithms & Comprehensions

3.1 MAML

Finn et al. [2017] proposed a method that can learn the parameters of any standard model via meta-learning in such a way as to prepare that model for fast adaptation. The intuition behind this approach is that some internal representations are more transferable than others. For example, a neural network might learn internal features that are broadly applicable to all tasks in $p(\mathcal{T})$, rather than a single individual task. The MAML meta-gradient update involves a gradient through a gradient. Computationally, this requires an additional backward pass through f to compute Hessian vector products, where f denotes the model that maps observations $\mathbf{x}$ to outputs $\mathbf{a}$. This backward pass is supported by standard deep learning API such as Torch [Paszke et al., 2019]. The full algorithm is given in Algorithm 1.

Algorithm 1 General Model-Agnostic Meta-Learning

Require $p(\mathcal{T})$: distribution over tasks;
Require α, β: step size hyper-parameters;
1. Randomly initialize θ

2. **while** *not done* **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$ **forall** $\mathcal{T}_i$ **do**
    1. Evaluate $\Delta_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    2. Compute adapted parameters with gradient descent: $\theta_i^{'} = \theta - \alpha \Delta_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end
  Update $\theta \leftarrow \theta - \beta \Delta_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i^{'}})$
end

There are many applications of this MAML algorithm, including reinforcement learning. The main differences between their applications are that they differ in the form of loss function and how the data is generated by the task. In this paper, our focus is on the area of reinforcement learning.

In particular, the goal of few-shot meta-learning is to enable an agent to quickly acquire a policy for a new test task using only a small amount of experience in the test setting. A new task might involve achieving a new goal or succeeding on a previously trained goal in a new environment. The MAML for reinforcement learning is given in Algorithm 2.

3.2 Taming MAML

One important fact overlooked in the original MAML method proposed by Finn et al. [2017] is the challenge of obtaining low variance and unbiased gradient estimates which is due to the difficulty of estimating the Hessian. In the original MAML implementation, the Hessian is approximated and gives a biased gradient estimate. Existing methods trying to solve the problem such as DICE Foerster et al. [2018] and LVC Rothfuss et al. [2018] still do not balance the trade-off between bias and variance well enough: DICE unbiased estimation suffers from high variance, LVC estimation has lower variance but is still biased. Thus, Taming MAML Liu et al. [2019] is proposed which is a surrogate objective function that adds control variates into gradient estimation via automatic differentiation. TMAML aims to improve the quality of gradient estimation by reducing

Algorithm 2 Model-Agnostic Meta-Learning for RL

Require $p(\mathcal{T})$: distribution over tasks;
Require α, β: step size hyper-parameters;
1. Randomly initialize θ

2. **while** *not done* **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$ **forall** $\mathcal{T}_i$ **do**
    1. Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
    2. Evaluate $\Delta_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$
    3. Compute adapted parameters with gradient descent: $\theta_i^{'} = \theta - \alpha \Delta_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    4. Sample trajectories $\mathcal{D}_i' = \{((\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta'}$ in $\mathcal{T}_i$
  end
  Update $\theta \leftarrow \theta - \beta \Delta_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$
end

variance without introducing bias: Improve estimation for the Hessian of the RL-objective by using a control variate, trained using meta-learning, which promotes better meta-policy gradient updates.

Liu et al. [2019] seek to learn policies that maximize cumulative expected rewards on a group of task-dependent MDPs and propose a surrogate function that can incorporate a control variate for each task by auto-differentiating twice, which reduce the variance of estimating the Hessian. The detailed proof can be found in Liu et al. [2019]. Given the unbiased estimates, meta control variate for each task which utilizes samples from other tasks is learned whose training process is described by the pseudo-code in Algorithm 3. The first-half trajectories are used to do an adaptation of meta control variates and then compute the inner loss of meta control variates on the other half of trajectories. Then outer gradient update is done using inner loss. Finally, the two groups of trajectories are swapped and repeat the same process until training loss converges.

Algorithm 3 Learning Meta Control Variates for TMAML

Require θ_V: meta control variates parameters;
Require α', β': step size hyper-parameters;
Require $\mathcal{D} = \{\tau_{1:N}\}$: episodes;

1. **while** θ_V *not converge* **do**
  Adapt θ_V with the first half of episodes $\{\tau_{1:N/2}\}$: $\theta_V^1 = \theta_V - \alpha' \nabla_{\theta_V} \mathcal{L}_{\theta_V}(\tau_{1:N/2})$
  Estimate meta control variates $V_{\theta_V^1}(s_t)$ for $s_t \sim \tau_{1:N/2}$ using adapted θ_V^1
  Adapt θ_V with the second half of episodes $\{\tau_{1:N/2}\}$: $\theta_V^2 = \theta_V - \alpha' \nabla_{\theta_V} \mathcal{L}_{\theta_V}(\tau_{N/2:N})$
  Estimate meta control variates $V_{\theta_V^2}(s_t)$ for $s_t \sim \tau_{1:N/2}$ using adapted θ_V^2
  Update meta control variates: $\theta_V \leftarrow \theta_V - \beta' \nabla_{\theta_V} \mathcal{L}_{\theta_V^1}(\tau_{N/2:N}) - \beta' \nabla_{\theta_V} \mathcal{L}_{\theta_V^2}(\tau_{1:N/2})$
  Swap trajectories $\{\tau_{1:N/2}\}$ and $\{\tau_{N/2:N}\}$
end

The pseudocode in Algorithm 4 combines TMAML and meta control variates together and describes the full training algorithm for optimizing θ to quickly adapt to new tasks.

3.3 PEARL [Seita and Rakelly, 2019]

The core of PEARL algorithm is to integrate the belief within the off-policy RL algorithm . To start, one can infer a variational approximation to the posterior belief with an encoder network $q_\phi(z|c)$ that takes context as input. To keep things tractable, PEARL represents the posterior as a Gaussian. For the RL agent, PEARL builds on Soft Actor-Critic (SAC) because of its state-of-the-art performance and sample efficiency. Samples from the belief are passed to the actor and critic so they can make predictions according to the sampled task. Meta-training then consists of learning to

Algorithm 4 Meta-RL with TMAML Gradient Estimators

Require $p(\mathcal{T})$: distribution over tasks ;
Require α, β: step size hyper-parameters;
1. Randomly initialize policy parameters θ and meta control variates parameters θ_V

2. **while** θ *not converge* **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$ *//Fitting control varirates* Learn meta control variates with Algorithm 3 **forall** $\mathcal{T}_i$ **do**
    //Inner gradient update θ 1. Sample pre-update episodes $\mathcal{D}_\rangle = \{\tau_i\}$ from $\mathcal{T}_i$ using π_θ
    2. Compute adapted parameters with gradient descent: $\theta^{'}_{\mathcal{T}_i} = \theta + \alpha\Delta_\theta(\mathcal{L}^{\text{TMAML}}_{\mathcal{T}_i}(\theta) + \mathcal{L}^{\text{DICE}}_{\mathcal{T}_i}(\theta))$ with $\mathcal{D}_\rangle = \{\tau_i\}$
    3. Sample post-update episodes $\mathcal{D}'_i = \{\tau'_i\}$ from $\mathcal{T}_i$ using $\pi_{\theta'}$
  end
  Update $\theta \leftarrow \theta + \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})}(\mathcal{L}^{\text{TMAML}}_{\mathcal{T}_i}(\theta'_{\mathcal{T}_i}) + \mathcal{L}^{\text{DICE}}_{\mathcal{T}_i}(\theta'_{\mathcal{T}_i}))$ using each $\mathcal{D}'_i = \{\tau'_i\}$
end

infer the posterior $q_\phi(z|c)$ given context and training the actor and critic to act optimally according to a given z. The encoder is optimized with gradients from the critic, so that $q_\phi(z|c)$ represents a distribution over Q-functions, as well as an information bottleneck. This bottleneck appears as a consequence of deriving the variational lower bound (ELBO).

Besides, the data batches are sampled to train the actor and critic which can be separated from the batches of context. Intuitively this makes much sense — by explicitly representing a belief over the task, the agent disentangles task inference from control and can use completely different data sources to learn each. This disentanglement is found empirically and heuristically to be important for off-policy meta-training. Heuristically, consider that meta-learning is predicated on the assumption that train and test time should match. For example, a meta-learner tasked with classifying mangoes and dragon fruits at test time should be trained on a distribution of classes that includes similar fruits. The analogy of this in RL is that if at test time the agent will adapt by gathering on-policy data, it should be trained with on-policy data as well. Using off-policy data during training thus presents a distribution shift that does not obey this basic assumption. In PEARL, one can mitigate this distribution shift while still using largely off-policy data by sampling on-policy data for the context, while using off-policy data for the actor-critic training.

The meta-training algorithm is given in Algorithm 5. The corresponding meta-testing procedure is to calculate the average return within a loop using the sampled data based on the parameters in the meta-training stage.

3.4 MACAW

Unlike existing meta-RL algorithms collects the data from the environment interactions online during meta-training, which is time-consuming. Besides, the existing data from other sources cannot be utilized by the online meta-RL algorithms, so the trained policy can be stuck in the sub-optimal policy. In order to cope with these problems, an offline meta-RL algorithm MACAW [Mitchell et al., 2021] is proposed, which combines MAML [Finn et al., 2017] with advantaged-weighted regression(AWR) and advantage regression loss and learn both value function V_ϕ and policy π_θ . Similar to MAML, MACAW meta-training contains inner-loop procedure, in which value function and policy update, and outer-loop procedure, which enables rapid adaption in the meta-test time. Since bootstrapping requires large iterations for estimating the value, the value function utilizes a bootstrap-free update

$$\mathcal{L}_V(\phi, D) = \mathbb{E}_{s,a\sim D}[(V_\phi(s) - \mathcal{R}_D(s,a))^2].$$

Since the gradient of the AWR object does not contain full information of both the regression weight and the regression target, the policy update includes the loss of AWR and advantage regression for

Algorithm 5 PEARL Meta-training

Require $\{\mathcal{T}_i\}_{i=1...T}$: batch of training tasks from $p(\mathcal{T})$;
Require $\alpha_1, \alpha_2, \alpha_3$: step size hyper-parameters;
1. Initialize replay buffers $\mathcal{B}^i$ for each training task

2. **while** *not done* **do**
  for each $\mathcal{T}_i$ **do**
    Initialize context $c^i = \{\}$
    for $k = 1, ..., K$ **do**
      Sample $z \sim q_\phi(z|c^i)$
      Gather data from $\pi_\theta(a|s, z)$ and add to $\mathcal{B}^i$
      Update $c^i = \left\{(s_j, a_j, s'_j, r_j)\right\}_{j=1,...,N} \sim \mathcal{B}^i$
    end
  end
  for step in training steps **do**
    for each $\mathcal{T}_i$ **do**
      Sample context $c^i \sim \mathcal{S}_c(\mathcal{B}^i)$ and RL batch $b^i \sim \mathcal{B}^i$
      Sample $z \sim q_\phi(z|c^i)$
      $\mathcal{L}^i_{actor} = \mathcal{L}_{actor}(b^i, z)$
      $\mathcal{L}^i_{critic} = \mathcal{L}_{critic}(b^i, z)$
      $\mathcal{L}^i_{KL} = \beta \mathcal{D}_{KL}(q(z|c^i)||r(z))$
    end
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \sum_i (\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL})$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \nabla_\theta \sum_i \mathcal{L}^i_{actor}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \nabla_\theta \sum_i \mathcal{L}^i_{critic}$
  end
end

the expressiveness of the policy update. Therefore, the loss function for the policy update is

$$\begin{aligned}\mathcal{L}_\pi(\theta, \phi'_i, D) =& \mathcal{L}^{\text{AWR}} + \lambda \mathcal{L}^{\text{ADV}} \\ =& \mathbb{E}_{s,a\sim D}[-\log \pi_{\theta(a|s)} \exp(\frac{1}{T}(\mathcal{R}_D(s,a) - V_{\phi'_i}(s)))] \\ &+ \lambda \mathbb{E}_{s,a\sim D}[(A_\theta(s,a) - (\mathcal{R}_D(s,a) - V_{\phi'_i}(s)))^2].\end{aligned}$$

However, only a standard advantage-weighted regression objective is adopted during the policy updating, since expressiveness concerns only pertain to the inner loop where only a small number of gradient steps are taken [Mitchell et al., 2021].

To summarize the aforementioned algorithms, we use the categories of 'policy updating' and 'data loading' to compare them, where the on-policy updating leverages the same ε-greedy policy that is evaluated and improved, which is also used to select actions. Off-policy instead is independent of the policy used during exploration. Online data loading means learning with new data coming in, while, offline means the training data is collected beforehand and is thus static. The comparison is given in Table 1. The summary of relationships between these algorithms is given in Figure 1.

4 Experiments & Results

4.1 Experiment settings

We test the performance of various meta-RL algorithms in three different tasks in MuJoCo [Todorov et al., 2012]: ant direction, half-cheetah direction, and half-cheetah velocity.

Algorithm 6 MACAW Meta-Training

Input: $\{\mathcal{T}_i\}$, offline buffers $\{\mathcal{D}_i\}$
Hyperparameters: learning rates $\alpha_1, \alpha_2, \eta_1, \eta_2$, training iterations n, temperature T;
1. Randomly initialize meta-parameters θ, ϕ

2. **for** n steps **do**
  for task $\mathcal{T}_i \in \mathcal{T}_i$ **do**
    Sample disjoint batches D_i^{tr}, $D_i^{ts} \sim D_i$
    $\phi_i' \leftarrow \phi - \eta_1 \Delta_\phi \mathcal{L}_V(\phi, D_i^{tr})$
    $\theta_i' \leftarrow \theta - \alpha_1 \Delta_\theta \mathcal{L}_\pi(\theta, \phi_i', D_i^{tr})$
  end
  $\phi \leftarrow \phi - \eta_2 \Delta_\phi \mathcal{L}_V(\phi_i', D_i^{ts})$
  $\theta \leftarrow \theta - \alpha_2 \Delta_\theta \mathcal{L}^{\text{AWR}}(\theta_i', \phi_i', D_i^{ts})$
end

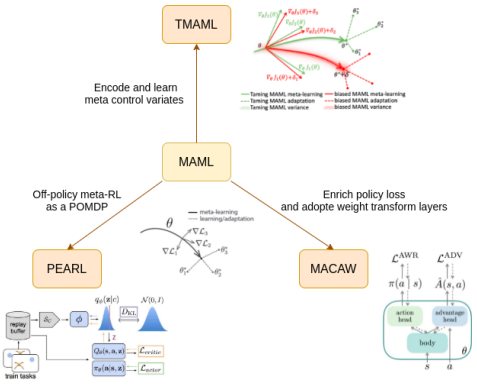


Figure 1: Relationship between the aforementioned algorithms

	Policy Updating	Data Loading	Notes
MAML	on-policy	online	model-agnostic framework
Taming MAML	on-policy	online	based on MAML, different obj. function
PEARL	off-policy	online	probabilistic & contextual method
MACAW	off-policy	offline	Combining MAML with off-policy value-based RL

Table 1: Comparisons between the aforementioned algorithms

For reproducing the result, while keeping the global environment settings unchanged, the following are some key hyperparameters corresponding to each algorithm:

1. MAML: "meta batch size": 40, "gamma": 0.99, "gae-lambda": 1, "fast-lr": 0.1, "num-steps": 1
2. Taming MAML: "meta batch size": 40, "discount": 0.99, "inner lr": 0.1, "learning rate": 0.001, "step size": 0.01
3. PEARL: "batch size": 256, "discount": 0.99, "policy lr": 3e-4, "qf lr":3e-4, "vf lr":3e-4, "context lr":3e-4
4. MACAW: "inner batch size": 32, "eval batch size": 32, "discount factor": 0.99, "inner policy lr": 0.01, "inner value lr": 0.01, "outer policy lr": 1e-3, "outer value lr": 1e-3

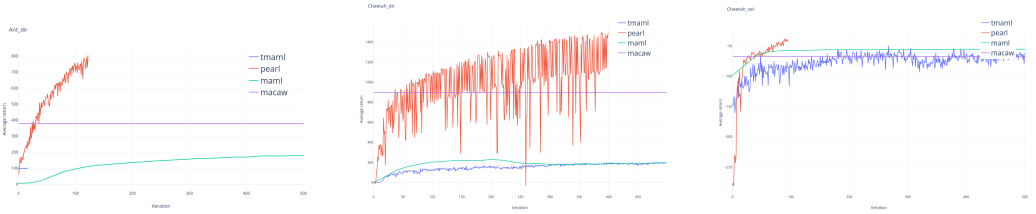
4.2 Experimental results

Our test tasks require two simulated robots – a planar 2D cheetah and a 3D ant – to run in a particular direction or at a particular velocity. In the velocity experiments, the reward is the negative absolute

value between the current velocity of the agent and a goal, which is chosen uniformly at random between 0.0 and 2.0 for the cheetah. In the direction experiments, the reward is the magnitude of the velocity in either the forward or backward direction, chosen at random for each task in $p(\mathcal{T})$.

We have tried to train each task using the aforementioned algorithms with as many episodes as possible. However, due to limited computational resources (GPU queuing), experiments of four algorithms ended up with different training iterations: MAML (10^3), Taming MAML (10^3), PEARL (10^2), MACAW (10^4). In Figure 2, we have only shown the first 500 episodes for sake of clarity. It is also worth noting that we choose to represent the reward of MACAW in a horizontal line using max avg. reward not only because the magnitude of its training iterations is quite large, but since the performance of MACAW is relatively inferior compared to others.

From Figure 2, we first noticed that Taming MAML has deficiencies while running the ant direction experiment, we'll further discuss this in the next section. Besides, we have noticed that although the PEARL performance seems not to converge, it still stands out among the four algorithms for all three tasks. Additionally, we want to clarify that the inner loop times of the PEARL are pretty large (10^4). This makes sense that PEARL can achieve the highest performance with fewer episodes. What's more, the other three algorithms seem to converge after a long training period. It is noticed that although Taming MAML claims that it uses a surrogate objective function of MAML, which at first glance is an improvement of MAML, the facts do not support this point at all. Finally, we want to point out that although PEARL's performances are excellent, however, in the half-cheetah direction experiment, it shows high volatility compared to others, which indicates that in some cases it is not robust enough to put into practice.



(a) 3D ant running in a fixed direction

(b) Half-cheetah running in a fixed direction

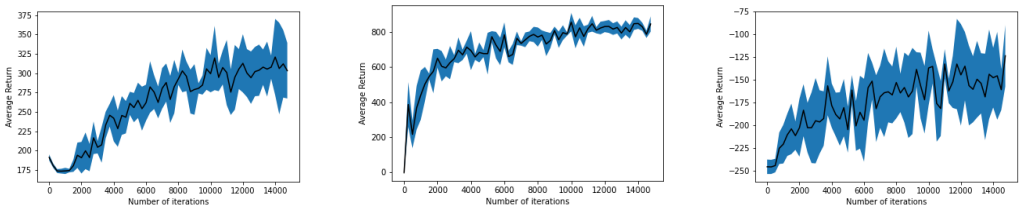
(c) Half-cheetah running in a fixed velocity

Figure 2: Comparison of meta-RL algorithms performance in three different scenarios with the purple horizontal line representing the maximal average reward by MACAW in each task

4.2.1 Special Cases in MACAW

Since the training steps (iterations) of offline meta-RL is fast compared to the online setting, it is by nature allowed us to run much more training iterations. Compared to 100 $\sim$ 1k iterations, we conducted 15k steps for the case of MACAW. To further test the robustness of the algorithm, we ran each experiment with six repetitions. As we can see from Figure 3, the left (ant with fixed direction) and the right (cheetah with fixed velocity) have unstable performances since when the iterations are approaching to finish, the averaged return is keeping increasing without any signal of convergence, and its variance is still large compared to the previous steps. Combined with the previous results, we accordingly doubt the author of the paper of MACAW who proposed that MACAW is superior to the algorithms of MAML and PEARL for the following reasons.

- Under the same environmental settings with different training hyperparameters, MACAW performs not as well as PEARL and MAML. However, the author intentionally compared it to the offline version of PEARL, which is meaningless from our point of view. The conclusion should be re-drawn with proper comparisons.
- The offline setting is fast since one has already pre-load the parameters. However, no matter it is fast or not, the most important thing is to actually "learn". From our repetition experiments, we didn't see any superior "learning" ability compared to the other three algorithms.



(a) 3D ant running in a fixed direction (b) 2D cheetah running in a fixed direction (c) 2D cheetah running in a fixed velocity

Figure 3: MACAW results for tasks with black line representing averaged return and blue region representing the standard deviation with six repetitions.

5 Future Work

As mentioned before, we have found that the ant direction experiment of Taming MAML is failed since it can only run for 17 training iterations. Though we have tried our best to debug and find the problem, it seems that the author does not configure the ant direction experiment correctly, the issue behind it is entangled between the version issue of Mujoco and the algorithm bugs. It is recommended that the authors of Taming MAML together with the community researchers figure out a bug-free open-source code repository.

Besides, it is also necessary to compare those algorithms in other tasks, such as those tasks from the Meta-World [Yu et al., 2019], to see if the PEARL algorithm still stands out among those algorithms. It is expected that the results would not be identical with high probability since different environments and task sets may impose more or fewer effects on the training process and thus rewards.

References

Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv:1611.05763 [cs, stat]*, January 2017. URL http://arxiv.org/abs/1611.05763. arXiv: 1611.05763.

Hao Liu, Richard Socher, and Caiming Xiong. Taming MAML: Efficient unbiased meta-reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4061–4071. PMLR, May 2019. URL https://proceedings.mlr.press/v97/liu19g.html. ISSN: 2640-3498.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]*, July 2017. URL http://arxiv.org/abs/1703.03400. arXiv: 1703.03400.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline Meta-Reinforcement Learning with Advantage Weighting. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7780–7791. PMLR, July 2021. URL https://proceedings.mlr.press/v139/mitchell21a.html. ISSN: 2640-3498.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. *arXiv:1903.08254 [cs, stat]*, March 2019. URL http://arxiv.org/abs/1903.08254. arXiv: 1903.08254.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*, 2018.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gescei. On the optimization of a synaptic learning rule. In *Optimality in Biological and Artificial Networks?*, pages 281–303. Routledge, 2013.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. Learning to explore via meta-policy gradient. In *International Conference on Machine Learning*, pages 5463–5472. PMLR, 2018a.

Flood Sung, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, 2017.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. *arXiv preprint arXiv:1805.09801*, 2018b.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients. *arXiv preprint arXiv:1510.09142*, 2015.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. *arXiv preprint arXiv:2002.02887*, 2020.

Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv preprint arXiv:1803.07246*, 2018.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *arXiv preprint arXiv:1506.05254*, 2015.

Théophane Weber, Nicolas Heess, Lars Buesing, and David Silver. Credit assignment techniques in stochastic computation graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2650–2660. PMLR, 2019.

Jingkai Mao, Jakob Foerster, Tim Rocktäschel, Maruan Al-Shedivat, Gregory Farquhar, and Shimon Whiteson. A baseline for any order gradient estimation in stochastic computation graphs. In *International Conference on Machine Learning*, pages 4343–4351. PMLR, 2019.

Joshua B Tenenbaum. Bayesian modeling of human concept learning. *Advances in neural information processing systems*, pages 59–68, 1999.

Li Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1134–1141. IEEE, 2003.