OFFLINE META-REINFORCEMENT LEARNING WITH ADVANTAGE WEIGHTING

Eric Mitchell* **Rafael Rafailov*** **Xue Bin Peng†** **Sergey Levine†** **Chelsea Finn***

ABSTRACT

Meta-reinforcement learning algorithms offer the promise of enabling rapid reinforcement learning of new skills by leveraging previous experience. However, while the adaptation phase of such methods is remarkably efficient, the meta-training phase of prior algorithms requires an impractical amount of experience to be collected in the loop of meta-training. Instead, we aim to enable batch offline meta-reinforcement learning — a class of algorithms that can meta-learn using only a batch of multi-task data to prepare for fast learning of new, related tasks. To this end, we develop an optimization-based meta-learning algorithm that uses simple supervised regression objectives for both inner-loop adaptation and outer-loop meta-learning, showing respectable performance on some common benchmarks and state-of-the-art performance on out-of-distribution tasks.

1 INTRODUCTION

Reinforcement learning is a powerful framework for autonomously acquiring skills such as game playing (Mnih et al., 2013) and robotic manipulation (Levine et al., 2016); but existing algorithms typically require millions of samples to learn a single task, making it impractical to learn a wide range of tasks. Meta-reinforcement learning is one promising approach to reduce the sample complexity of training (Duan et al., 2016; Wang et al., 2016; Finn et al., 2017). Meta-RL algorithms aim to exploit shared structure within a family of tasks to amortize the cost of learning across them, instead of learning each task separately from scratch. In this work, we aim to develop an offline meta-reinforcement learning algorithm that is sample efficient during meta-training, that can make use of previous offline data, and that produces an adaptation procedure that is effective both within and outside of the meta-training task distribution.

To this end, we propose an algorithm that meta-trains using arbitrary offline (or batch) experience from previous tasks, enabling efficient transfer to new tasks without any interaction with the meta-training environments. We consider this setting in which an agent attempts to meta-learn from pre-existing, offline data, without further interactions, as the *offline meta-RL* problem. Because this problem setting places no constraint on the distribution of data collected for training, we expect that this problem setting will be much more practically useful than the standard online meta-RL setting, as it permits meta-training from arbitrary data sources. Further, this problem setting is analogous to the widely-adopted pre-training and fine-tuning paradigm in supervised learning, where pre-training can be performed using an offline dataset of previously-collected experience.

To address this problem setting, we aim to extend the model-agnostic meta-learning algorithm (Finn et al., 2017), since it directly corresponds to fine-tuning at meta-test time, which facilitates out-of-distribution robustness (Finn & Levine, 2017). However, the existing MAML algorithm uses online policy gradients, which cannot be applied to the offline setting and is especially data hungry. Further, combining MAML with value-based RL subroutines is not straightforward: the higher-order optimization in MAML-like methods demands stable and efficient gradient-descent updates, while TD backups are both somewhat unstable and require a large number of steps. We address this issue by utilizing a supervised bootstrap-free subroutine (Peters & Schaal, 2007; Peng et al., 2019) for both for the inner and outer loop of a gradient-based meta-learning algorithm. Our method adapts both a stationary value function, via Monte Carlo returns, and a policy, via weighted regression using the adapted value function. Further, these updates can all be performed in the offline setting, while the resulting test time adaptation corresponds to *consistent* optimization procedure that is robust to out-of-distribution tasks.

*Stanford University †UC Berkeley Correspondence: eric.mitchell@cs.stanford.edu

Algorithm 1 MACAW Meta-Training

Input: Tasks $\{\mathcal{T}_i\}$, offline trajectory buffers $\{\mathcal{D}_i\}$
Hyperparameters: learning rates α_1, α_2, η_1, η_2, training iterations n, temperature T
Randomly initialize meta-parameters θ, ϕ
for n steps **do**
  for task $\mathcal{T}_i \in \{\mathcal{T}_i\}$ **do**
    Sample disjoint batches $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{ts}} \sim \mathcal{D}_i$
    $\phi_i' \leftarrow \phi - \eta_1 \nabla_\phi \mathcal{L}_V(\phi, \mathcal{D}_i^{\text{tr}})$
    $\theta_i' \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi(\theta, \phi_i', \mathcal{D}_i^{\text{tr}})$
  $\phi \leftarrow \phi - \eta_2 \sum_i \left[\nabla_\phi \mathcal{L}_V(\phi_i', \mathcal{D}_i^{\text{ts}})\right]$
  $\theta \leftarrow \theta - \alpha_2 \sum_i \left[\nabla_\theta \mathcal{L}^{\text{AWR}}(\theta_i', \phi_i', \mathcal{D}_i^{\text{ts}})\right]$

Algorithm 2 MACAW Meta-Testing

Input: Test task $\mathcal{T}_j$, offline experience $\mathcal{D}_j$, meta-policy π_θ, meta-value function V_ϕ
Hyperparameters: learning rates α, η, adaptation iterations n, temperature T
Initialize $\theta_0 \leftarrow \theta$, $\phi_0 \leftarrow \phi$.
for n steps **do**
  $\phi_{t+1} \leftarrow \phi_t - \eta_1 \nabla_{\phi_t} \mathcal{L}_V(\phi_t, \mathcal{D}_j)$
  $\theta_{t+1} \leftarrow \theta_t - \alpha_1 \nabla_{\theta_t} \mathcal{L}_\pi(\theta_t, \phi_{t+1}, \mathcal{D}_j)$

2 Preliminaries and Problem Statement

Meta-Reinforcement Learning. Typical formulations of meta-RL consider tasks drawn from a distribution over tasks, $\mathcal{T} \sim p(\mathcal{T})$, where each task is an MDP. All tasks within a family are generally assumed to possess some shared structure enabling amortization of the cost of learning across different tasks in the family. During training, an agent is presented with different tasks from sampled from $p(\mathcal{T})$; at test time, an agent's objective is to rapidly find a high-performing policy for a (potentially unseen) task $\mathcal{T}' \sim p(\mathcal{T})$. That is, with only a small amount of experience on $\mathcal{T}'$, the agent should find a policy that achieves high expected reward on that task. During meta-training, the agent meta-learns a set of parameters and/or update rule that enables such rapid adaptation at test-time, typically by training on a set of training tasks drawn from $p(\mathcal{T})$ but not including the tasks seen at test time.

The Offline Meta-RL Problem. In the offline meta-RL problem setting, we aim to leverage offline multi-task experience to enable fast adaptation to new downstream tasks. We assume that tasks $\mathcal{T}_i$ are drawn from a family of tasks $\mathcal{T}$ according to some distribution $p(\mathcal{T})$. Each task $\mathcal{T}_i$ is a Markov decision process tuple $(\mathcal{S}, \mathcal{A}, p_i, r_i)$. In the offline setting, a task also has an associated fixed dataset $\mathcal{D}_i$, from which experiences on the task are drawn. Sampling data from a fixed dataset, rather than from the environment, distinguishes offline meta-RL from the standard meta-RL setting. Like standard meta-RL, offline meta-RL has a meta-training and meta-testing phase; during meta-training, an agent trains on a dataset $\{\mathcal{D}_i\}$ from each of a set of training tasks $\{\mathcal{T}_i\}$. Unlike in the standard meta-RL setting, in offline meta-RL, the agent is not able to collect any additional experience from the environment according to its own policy. During meta-testing, a test task $\mathcal{T}_{\text{test}}$ is drawn from $p(\mathcal{T})$, and the meta-trained agent is presented with a new batch of experience $\mathcal{B}_{\text{test}}$ sampled from the distribution $\mathcal{D}_{\text{test}}$. The agent's objective is to use this batch of adaptation to find the highest-performing policy for the test task.

3 MACAW: Meta Actor-Critic with Advantage Weighting

The offline meta-reinforcement learning setting inherits the distributional difficulties of offline RL: the agent is not able to control the data distribution, an option that prior meta-RL methods require (Finn et al., 2017; Rakelly et al., 2019). In addition to satisfying the requirements of this problem, an ideal method should be *consistent* even with out-of-distribution tasks, in the sense that given sufficient adaptation data at meta-test time, the algorithm should find a solution to that task, to the degree to which the data permits. To address these challenges, we propose meta actor-critic with advantage weighting (MACAW), which we describe in this section.

3.1 Adaptation Procedure for MACAW

At a high level, the adaptation process will correspond to a value function update followed by a policy update. Optimization-based meta-learning methods typically rely on truncated optimizations for the adaptation process (Finn et al., 2017), to satisfy both computational and memory constraints (Wu et al., 2018; Rajeswaran et al., 2019). We will also use a truncated optimization. However, value-based RL algorithms such as Q-learning require many updates for values to propagate. Motivated by this challenge, we choose to use a light-weight value estimation procedure, based on supervised Monte Carlo returns, placing greater burden on the policy update that leverages the estimated value function.

In particular, we will assume a batch of training data $\mathcal{D}_i^{\text{tr}}$ for adapting to $\mathcal{T}_i$. MACAW updates the value function by taking one or a few gradient steps on a supervised Monte Carlo objective as follows:

$$\phi_i' \leftarrow \phi - \eta_1 \nabla_\phi \mathcal{L}_V(\phi, \mathcal{D}_i^{\text{tr}}), \quad \text{where} \quad \mathcal{L}_V(\phi, \mathcal{D}) \triangleq \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[(V_\phi(\mathbf{s}) - \mathcal{R}_\mathcal{D}(\mathbf{s}))^2\right], \tag{1}$$

and where $\mathcal{R}_\mathcal{D}(\mathbf{s})$ is the Monte Carlo return from the state $\mathbf{s}$ observed from that roll-out in $\mathcal{D}$.

The AWR algorithm updates its policy by performing supervised regression onto actions weighted by the estimated advantage. While it is tempting to use this same update rule here, we note that this update does not provide the meta-learner with significant expressive power. For MAML-based methods to approximate any learning procedure, the gradient must not discard information needed to infer the task Finn & Levine (2017). Furthermore, the gradient of the advantage-weighted regression objective does not contain full information of both the regression weight and the regression target. That is, you cannot recover both the advantage weight and the action from the gradient. To address this issue and make our meta-learner sufficiently expressive, our policy update will perform both advantage-weighted regression onto actions and an auxiliary regression onto advantages.

Concretely, our policy architecture has two heads corresponding to the predicted action given state, $\pi_\theta(\cdot|\mathbf{s})$, and the predicted advantage given both state and action $\pi_\theta(\cdot|\mathbf{s}, \mathbf{a})$. Policy adaptation proceeds as follows:

$$\theta_i' \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi(\theta, \phi_i', \mathcal{D}_i^{\text{tr}}), \quad \text{where} \quad \mathcal{L}_\pi = \mathcal{L}^{\text{AWR}} + \lambda \mathcal{L}^{\text{ADV}}. \tag{2}$$

In our policy update, we show only one gradient step for conciseness of notation, but using multiple gradient steps is straightforward. The AWR loss with temperature T corresponds to:

$$\mathcal{L}^{\text{AWR}}(\theta, \phi_i', \mathcal{D}) \triangleq \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[-\log \pi_\theta(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{T}\left(\mathcal{R}_\mathcal{D}(\mathbf{s}, \mathbf{a}) - V_{\phi_i'}(\mathbf{s})\right)\right)\right], \tag{3}$$

and the advantage regression loss can be written as:

$$\mathcal{L}^{\text{ADV}}(\theta, \phi_i', \mathcal{D}) \triangleq \mathop{\mathbb{E}}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[(\pi_\theta(\cdot|\mathbf{s}, \mathbf{a}) - \left(\mathcal{R}_\mathcal{D}(\mathbf{s}, \mathbf{a}) - V_{\phi_i'}(\mathbf{s})\right))^2\right] \tag{4}$$

This adaptation procedure is summarized in Algorithm 2. Next, we describe how we can optimize for meta-parameters θ and ϕ, i.e. the initialization of the policy and value respectively, such that this adaptation is fast and effective.

3.2 Outer Loop MACAW Optimization

To enable rapid policy adaptation at meta-test time, we meta-train for a set of initial parameters of both the value function and policy. To do so, we use two disjoint sets of offline data, $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{ts}}$, for each task $\mathcal{T}_i$, and use these datasets for the inner and outer loops respectively. By using disjoint datasets, we encourage few-shot generalization from the training data $\mathcal{D}_i^{\text{tr}}$ rather than memorization.

The value function meta-learning procedure follows MAML, but with the supervised Monte Carlo objective:

$$\min_\phi \mathbb{E}_{\mathcal{T}_i}\left[\mathcal{L}_V(\phi_i', \mathcal{D}_i^{\text{ts}})\right] = \min_\phi \mathbb{E}_{\mathcal{T}_i}\left[\mathcal{L}_V(\phi - \eta_1 \nabla_\phi \mathcal{L}_V(\phi, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})\right] \tag{5}$$

That is, we optimize for a set of initial value function parameters such that one or a few inner gradient steps leads to an accurate value estimate, where $\mathcal{L}_V$ is defined in Equation 1.

Since expressivity concerns only pertain to the learned inner loop, the outer loss for the policy consist of a vanilla AWR objective:

$$\min_\theta \mathbb{E}_{\mathcal{T}_i}\left[\mathcal{L}^{\text{AWR}}(\theta_i', \phi_i', \mathcal{D}_i^{\text{ts}})\right] = \min_\theta \mathbb{E}_{\mathcal{T}_i}\left[\mathcal{L}^{\text{AWR}}(\theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi(\theta, \phi_i', \mathcal{D}_i^{\text{tr}}), \phi_i', \mathcal{D}_i^{\text{ts}})\right], \tag{6}$$

where $\mathcal{L}_\pi$ is defined in Equation 2 and $\mathcal{L}^{\text{AWR}}$ is defined in Equation 3. Note that we use the adapted value function parameters in all cases. The meta-training algorithm is summarized in Algorithm 1.

4 Experiments

The goal of our empirical evaluation is to first and foremost answer our key scientific hypothesis: can we acquire priors from offline multi-task data that facilitate transfer to new tasks? Further, how does our approach's performance compare to that of previously-proposed algorithms such as

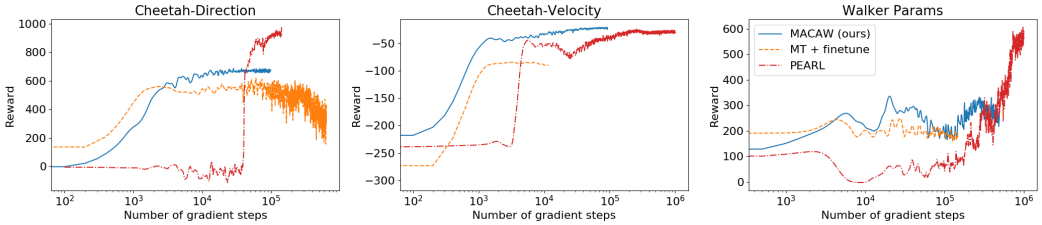

Figure 1: **Offline Meta-RL results**: We compare MACAW with PEARL (Rakelly et al., 2019), a state-of-the-art off-policy meta-RL method, and a multi-task training + fine tuning baseline, which is allowed to take 20 gradient steps during each evaluation to 'adapt' to each test task. In contrast, the presented performance for MACAW is with a single gradient step of adaptation. We find that MACAW learns more quickly than PEARL, though not always achieving the same level of asymptotic performance. (Note that the x-axis is in log scale.)

PEARL (Rakelly et al., 2019) and fine-tuning, and how does our method perform in the setting of out-of-distribution test tasks?

Tasks and Experimental Set-Up. We base our experiments on several environments used in previous meta-RL works (Finn et al. (2017); Rakelly et al. (2019)), in particular, the Half-Cheetah direction and velocity environments and the Walker2d with random parameters environment. More detailed description is given in the appendix.

Experimental Results Figure 1 shows the results of our comparison between MACAW, PEARL, and multi-task pre-training and fine-tuning, illustrating test task performance over the course of meta-training. We observe that MACAW is indeed able to learn from fully offline data, and consistently outperforms multi-task training and fine-tuning. Surprisingly, we find that PEARL also achieves good performance in this setting, despite its design as an off-policy meta-RL method. Although MACAW does not always achieve the same asymptotic performance as PEARL, it generally learns with fewer gradient steps and finds a respectable solution. However, we find that MACAW is substantially more performant when faced with out-of-distribution tasks; Figure 2 shows adaptation performance on out-of-distribution tasks as a function of the number of test time gradient steps. As expected, since MACAW recovers a well-formed optimization at adaptation time, it significantly outperforms PEARL. We note that in this experiment, MACAW and MT + fine tune adapt at test time with 8 samples, while PEARL is given 256 samples from its context buffer. Even with far fewer context samples to adapt from, MACAW generally finds significantly better solutions than PEARL.

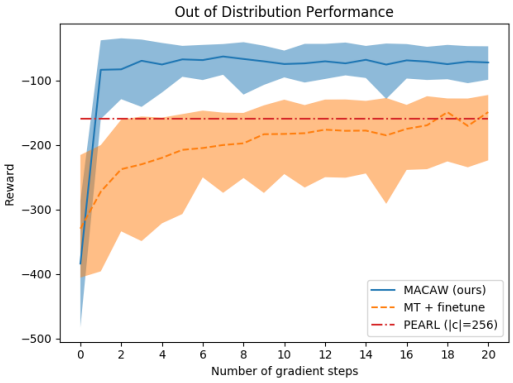

Figure 2: **Performance on Out-of-Distribution Tasks.** Median performance of MACAW, PEARL, and the multi-task + fine tuning comparison on out-of-distribution tasks in the Cheetah Velocity environment. Error bars correspond to the 25th and 75th percentile. We find that MACAW is significantly more robust to task distribution shift.

5 Conclusion

In this work, we formulated the problem of offline meta-reinforcement learning and presented MACAW, a practical algorithm that achieves good performance on various continuous control tasks compared to other state-of-the-art meta-RL algorithms. We motivated the design of MACAW by the desire to build a meta-RL algorithm that is both sample-efficient (using value-based RL subroutines) and consistent (running a full-fledged RL algorithm at test time). We hope that this work serves as the basis for future research in offline meta-RL, enabling more sample-efficient learning algorithms to make better use of purely observational data and adapt effectively to new environments.

REFERENCES

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. 10 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 745–750, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273590. URL https://doi.org/10.1145/1273496.1273590.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pp. 113–124, 2019.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, 2019.

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 preprint arXiv:1611.05763*, 2016.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018.

A APPENDIX

A.1 EXPERIMENTAL DETAILS

For the Half-Cheetah-velocity environment we sample 40 tasks with target speeds in the interval $[0, 3]$. For our main comparison shown in Figure 1, we train on a random selection of 35 of those tasks and use the remaining 5 as our meta-test set. For the out-of-distribution experiment (shown in Figure 2) we use 13 training tasks with speeds in the interval [0,1], and 27 test with target velocities within [1,3]. On the Walker2d environment, we randomly sample 35 tasks and use a random 30/5 meta-train/test split. We adapt each of these problems to the offline setting by restricting the data to a fixed offline buffer of 1 million transitions, gathered from the lifetime of an agent trained with Soft Actor Critic (Haarnoja et al., 2018a;b) on each of the tasks of interest. The buffer for offline adaptation at meta-test time is a small random sample of the corresponding test task data.

A.2 Environment Descriptions

The problems of interest include:

1. *Half-Cheetah Direction* Train a simple cheetah to run in one of two direction: forward and backward. In other words, the space of task parameters ψ include only two values, the two possible target directions 1 and -1. The meta-test tasks are the same as the meta-train.
2. *Half-Cheetah Velocity* Train a cheetah to run at a desired velocity, which fully parameterizes each task. For our experiments, values of the task parameters are sampled from a uniform interval of 40 velocities in the range [0, 3]. Held out test target velocities are sampled from the same set.
3. *Walker-2D Params* Train a simulated agent to move forward, where different tasks correspond to different randomized environment dynamics parameters rather than reward functions.