Offline Meta-Reinforcement Learning with Advantage Weighting

Eric Mitchell [1] Rafael Rafailov [1] Xue Bin Peng [2] Sergey Levine [2] Chelsea Finn [1]

Abstract

This paper introduces the offline meta-reinforcement learning (offline meta-RL) problem setting and proposes an algorithm that performs well in this setting. Offline meta-RL is analogous to the widely successful supervised learning strategy of pre-training a model on a large batch of fixed, pre-collected data (possibly from various tasks) and fine-tuning the model to a new task with relatively little data. That is, in offline meta-RL, we meta-train on fixed, pre-collected data from several tasks in order to adapt to a new task with a very small amount (less than 5 trajectories) of data from the new task. By nature of being offline, algorithms for offline meta-RL can utilize the largest possible pool of training data available and eliminate potentially unsafe or costly data collection during meta-training. This setting inherits the challenges of offline RL, but it differs significantly because offline RL does not generally consider a) transfer to new tasks or b) limited data from the test task, both of which we face in offline meta-RL. Targeting the offline meta-RL setting, we propose Meta-Actor Critic with Advantage Weighting (MACAW), an optimization-based meta-learning algorithm that uses simple, supervised regression objectives for both the inner and outer loop of meta-training. On offline variants of common meta-RL benchmarks, we empirically find that this approach enables fully offline meta-reinforcement learning and achieves notable gains over prior methods. Code available at https://sites.google.com/view/macaw-metarl.

1. Introduction

Meta-reinforcement learning (meta-RL) has emerged as a promising strategy for tackling the high sample complexity of reinforcement learning algorithms, when the goal is to ultimately learn many tasks. Meta-RL algorithms exploit shared structure among tasks during meta-training, amortizing the cost of learning across tasks and enabling rapid adaptation to new tasks during meta-testing from only a small amount of experience. Yet unlike in supervised learning, where large amounts of pre-collected data can be pooled from many sources to train a single model, existing meta-RL algorithms assume the ability to collect millions of environment interactions *online* during meta-training. Developing *offline* meta-RL methods would enable such methods, in principle, to leverage existing data from any source, making them easier to scale to real-world problems where large amounts of data might be necessary to generalize broadly. To this end, we propose the offline meta-RL problem setting and a corresponding algorithm that uses only offline (or batch) experience from a set of training tasks to enable efficient transfer to new tasks without requiring any further interaction with *either* the training or testing environments. See Figure 1 for a comparison of offline meta-RL and standard meta-RL.

Because the offline setting does not allow additional data collection during training, it highlights the desirability of a *consistent* meta-RL algorithm. A meta-RL algorithm is consistent if, given enough diverse data on the test task, adaptation can find a good policy for the task regardless of the training task distribution. Such an algorithm would provide a) rapid adaptation to new tasks from the same distribution as the train tasks while b) allowing for improvement even for out of distribution test tasks. However, designing a consistent meta-RL algorithm in the offline setting is difficult: the consistency requirement suggests we might aim to extend the model-agnostic meta-learning (MAML) algorithm (Finn et al., 2017a), since it directly corresponds to fine-tuning at meta-test time. However, existing MAML approaches use online policy gradients, and only value-based approaches have proven effective in the offline setting. Yet 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 to propagate reward information across long time horizons.

To address these challenges, one might combine MAML

[1]Stanford University [2]University of California, Berkeley. Correspondence to: Eric Mitchell <eric.mitchell@cs.stanford.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

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. However, as we will discuss in Section 4 and find empirically in Section 5, naïvely combining MAML and AWR in this way does not provide satisfactory performance because the AWR policy update is not sufficiently expressive. Motivated by prior work that studies the expressive power of MAML (Finn and Levine, 2018), we increase the expressive power of the meta-learner by introducing a carefully chosen policy update in the inner loop. We theoretically prove that this change increases the richness of the policy's update and find empirically that this policy update can dramatically improve adaptation performance and stability in some settings. We further observe that the relatively shallow feedforward neural network architectures used in reinforcement learning are not well-suited to optimization-based meta-learning and suggest an alternative that proves critical for good performance across four different environments. We call the resulting meta-RL algorithm and architecture meta actor-critic with advantage weighting, or MACAW.

Our main contributions are the offline meta-RL problem setting itself and MACAW, an offline meta-reinforcement learning algorithm and architecture that possesses three key properties: sample efficiency, offline meta-training, and consistency at meta-test time. To our knowledge, MACAW is the first algorithm to successfully combine gradient-based meta-learning and off-policy value-based RL. Our evaluations include experiments on offline variants of standard continuous control meta-RL benchmarks as well as settings specifically designed to test the robustness of an offline meta-learner with scarce or poor-quality training data. In all comparisons, MACAW significantly outperforms fully offline variants state-of-the-art off-policy RL and meta-RL baselines.

2. Preliminaries

In reinforcement learning, an agent interacts with a Markov Decision Process (MDP) to maximize its cumulative reward. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, T, r)$ consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, stochastic transition dynamics $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and a reward function r. At each time step, the agent receives reward $r_t = r(s_t, a_t, s_{t+1})$. The agent's objective is to maximize the expected return (i.e. discounted sum of rewards) $\mathcal{R} = \sum_t \gamma^t r_t$, where $\gamma \in [0, 1]$ is a discount factor. Existing meta-RL problem statements generally consider tasks drawn from a distribution $\mathcal{T}_i \sim p(\mathcal{T})$, where each task $\mathcal{T}_i = (\mathcal{S}, \mathcal{A}, p_i, r_i)$ represents a different MDP. Both the dynamics and reward function may vary across tasks. The tasks are generally assumed to exhibit some (unknown) shared structure. During meta-training, the agent is presented with tasks sampled from $p(\mathcal{T})$; at meta-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 return on that task. During meta-training, the agent meta-learns parameters or update rules that enables such rapid adaptation at test-time.

Model-agnostic meta-learning One class of algorithms for addressing the meta-RL problem (as well as meta-supervised learning) are variants of the MAML algorithm (Finn et al., 2017a), which involves a bi-level optimization that aims to achieve fast adaptation via a few gradient updates. Specifically, MAML optimizes a set of initial policy parameters θ such that a few gradient-descent steps from θ leads to policy parameters that achieve good task performance. At each meta-training step, the inner loop adapts θ to a task $\mathcal{T}$ by computing $\theta' = \theta - \alpha \nabla_\theta \mathcal{L}_\mathcal{T}(\theta)$, where $\mathcal{L}$ is the loss function for task $\mathcal{T}$ and α is the step size (in general, θ' might be computed from multiple gradient steps, rather than just one as is written here). The outer loop updates the initial parameters as $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}'_\mathcal{T}(\theta')$, where $\mathcal{L}'_\mathcal{T}$ is a loss function for task $\mathcal{T}$, which may or may not be the same as the inner-loop loss function $\mathcal{L}_\mathcal{T}$, and β is the outer loop step size. MAML has been previously instantiated with policy gradient updates in the inner and outer loops (Finn et al., 2017a; Rothfuss et al., 2018), which can only be applied to on-policy meta-RL settings; we address this shortcoming in this work.

Advantage-weighted regression. To develop an offline meta-RL algorithm, we build upon advantage-weighted regression (AWR) (Peng et al., 2019), a simple offline RL method. The AWR policy objective is given by

$$\mathcal{L}^{\text{AWR}}(\vartheta, \varphi, B) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim B}\left[-\log \pi_\vartheta(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{T} \left(\mathcal{R}_B(\mathbf{s}, \mathbf{a}) - V_\varphi(\mathbf{s}) \right) \right) \right] \tag{1}$$

where $B = \{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ can be an arbitrary dataset of transition tuples sampled from some behavior policy, and $\mathcal{R}_B(\mathbf{s}, \mathbf{a})$ is the return recorded in the dataset for performing action $\mathbf{a}$ in state $\mathbf{s}$, $V_\varphi(\mathbf{s})$ is the learned value function for the behavior policy evaluated at state $\mathbf{s}$, and $T > 0$ is a temperature parameter. The term $\mathcal{R}_B(\mathbf{s}, \mathbf{a}) - V_\varphi(\mathbf{s})$ represents the advantage of a particular action. The objective can be interpreted as a weighted regression problem, where actions that lead to higher advantages are assigned larger weights. The value function parameters φ are typically trained using simple regression onto Monte Carlo returns, and the policy parameters ϑ are trained using $\mathcal{L}^{\text{AWR}}$. Next, we discuss the offline meta-RL problem and some of the challenges it poses.

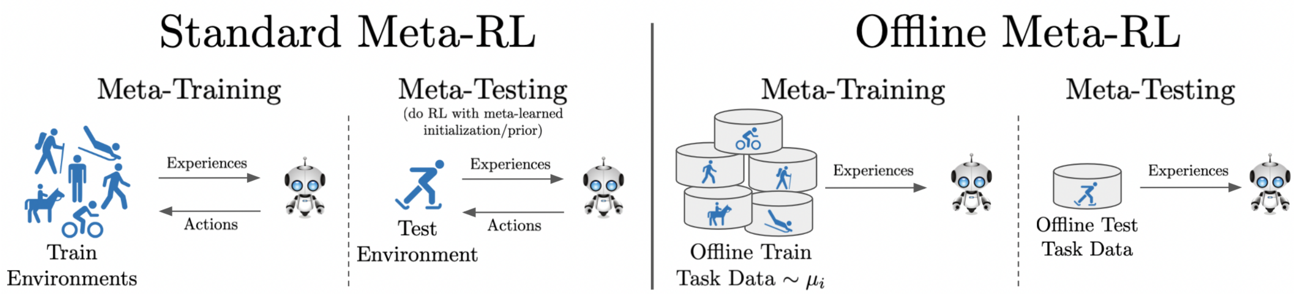

Figure 1: Comparing the standard meta-RL setting (left), which includes on-policy and off-policy meta-RL, with offline meta-RL (right). In standard meta-RL, new interactions are sampled from the environment during both meta-training and meta-testing, potentially storing experiences in a replay buffer (off-policy meta-RL). In *offline* meta-RL, a batch of data is provided for each training task $\mathcal{T}_i$. This data could be the result of prior skills learned, demonstrations, or other means of data collection. The meta-learner uses these static buffers of data for meta-training and can then learn a new test task when given a small buffer of data for that task.

3. 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. A task $\mathcal{T}_i$ is defined as a tuple $(\mathcal{M}_i, \mu_i)$ containing a Markov decision process (MDP) $\mathcal{M}_i$ and a fixed, unknown behavior policy μ_i. Each μ_i might be an expert policy, sub-optimal policy, or a mixture policy (e.g. corresponding the replay buffer of an RL agent). Tasks are drawn from a task distribution $p(\mathcal{T}) = p(\mathcal{M}, \mu)$. During meta-training, an offline meta-RL algorithm cannot interact with the environment, and instead has access only to a fixed buffer of transition tuples $D_i = \{s_{i,j}, a_{i,j}, s'_{i,j}, r_{i,j}\}$ sampled from μ_i for each task. During meta-testing, a (typically unseen) test task $\mathcal{T}_{\text{test}} = (\mathcal{M}_{\text{test}}, \mu_{\text{test}})$ is drawn from $p(\mathcal{T})$. We consider two different meta-testing procedures. In the *fully offline meta-RL* setting, the meta-trained agent is presented with a small batch of experience D_{test} sampled from μ_{test}. The agent's objective is to adapt using *only* D_{test} to find the highest-performing policy possible for $\mathcal{M}_{\text{test}}$. Alternatively, in the *offline meta-RL with online fine-tuning* setting, the agent can perform additional online data collection and learning after being provided with the offline data D_{test}. Note that in both settings, if μ_{test} samples data uninformative for solving $\mathcal{M}_{\text{test}}$, we might expect test performance to be affected; we consider this possibility in our experiments.

Sampling data from fixed batches at meta-training time, rather than from the learned policy itself, distinguishes offline meta-RL from the standard meta-RL setting. This setting is particularly applicable in situations when allowing online exploration might be difficult, expensive, or dangerous. We introduce both variants of the problem because for some settings, limited online data collection may be possible, especially with a reasonably performant/safe policy acquired through fully offline adaptation. Prior meta-RL methods require interaction with the MDP for each of the meta-training tasks (Finn et al., 2017a), 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, in addition to satisfying the consistency desideratum described in Section 1.

4. MACAW: Meta Actor-Critic with Advantage Weighting

To address the numerous challenges posed by offline meta-RL, we propose meta actor-critic with advantage weighting (MACAW). MACAW is an offline meta-RL algorithm that learns initializations ϕ and θ for a value function V_ϕ and policy π_θ, respectively, that can rapidly adapt to a new task seen at meta-test time via gradient descent. Both the value function and the policy objectives correspond to simple weighted regression losses in both the inner and outer loop, leading to a stable and consistent inner-loop adaptation process and outer-loop meta-training signal. While these objectives build upon AWR, we show that the naive application of an AWR update in the inner loop can lead to unsatisfactory performance, motivating the enriched policy update that we describe in Section 4.1. In Sections 4.2 and 4.3, we detail the full meta-training procedure and an important architectural component of the policy and value networks.

Algorithm 1 MACAW Meta-Training

1: **Input:** Tasks $\{\mathcal{T}_i\}$, offline buffers $\{D_i\}$
2: **Hyperparameters**: learning rates $\alpha_1, \alpha_2, \eta_1, \eta_2$, training iterations n, temperature T
3: Randomly initialize meta-parameters θ, ϕ
4: **for** n steps **do**
5:     **for** task $\mathcal{T}_i \in \{\mathcal{T}_i\}$ **do**
6:         Sample disjoint batches $D_i^{\text{tr}}, D_i^{\text{ts}} \sim D_i$
7:         $\phi_i' \leftarrow \phi - \eta_1 \nabla_\phi \mathcal{L}_V(\phi, D_i^{\text{tr}})$
8:         $\theta_i' \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi(\theta, \phi_i', D_i^{\text{tr}})$
9:     **end for**
10:     $\phi \leftarrow \phi - \eta_2 \sum_i \left[\nabla_\phi \mathcal{L}_V(\phi_i', D_i^{\text{ts}})\right]$
11:     $\theta \leftarrow \theta - \alpha_2 \sum_i \left[\nabla_\theta \mathcal{L}^{\text{AWR}}(\theta_i', \phi_i', D_i^{\text{ts}})\right]$
12: **end for**

Algorithm 2 MACAW Meta-Testing

Input: Test task $\mathcal{T}_j$, offline experience D, meta-policy π_θ, meta-value function V_ϕ
Hyperparameters: learning rates α_1, η, adaptation iterations n, temperature T
Initialize $\theta_0 \leftarrow \theta$, $\phi_0 \leftarrow \phi$.
for n steps **do**
    $\phi_{t+1} \leftarrow \phi_t - \eta_1 \nabla_{\phi_t} \mathcal{L}_V(\phi_t, D)$
    $\theta_{t+1} \leftarrow \theta_t - \alpha_1 \nabla_{\theta_t} \mathcal{L}_\pi(\theta_t, \phi_{t+1}, D)$
end for

4.1. Inner-Loop MACAW Procedure

The adaptation process for MACAW consists of a value function update followed by a policy update and can be found in lines 6-8 in Algorithm 1. Optimization-based meta-learning methods typically rely on truncated optimization for the adaptation process (Finn et al., 2017a), to satisfy both computational and memory constraints (Wu et al., 2018; Rajeswaran et al., 2019), and MACAW also uses a truncated optimization. However, value-based algorithms that use bootstrapping, such as Q-learning, can require many iterations for values to propagate. Therefore, we use a bootstrap-free update for the value function that simply performs supervised regression onto Monte-Carlo returns. Given a batch of training data D_i^{tr} collected for $\mathcal{T}_i$, MACAW adapts the value function by taking one or a few gradient steps on the following supervised objective:

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

and where $\mathcal{R}_D(\mathbf{s},\mathbf{a})$ is the Monte Carlo return from the state $\mathbf{s}$ taking action $\mathbf{a}$ observed in D.

After adapting the value function, we proceed to adapting the policy. The AWR algorithm updates its policy by performing supervised regression onto actions weighted by the estimated advantage, where the advantage is given by the return minus the value: $\mathcal{R}_D(\mathbf{s},\mathbf{a}) - V_{\phi_i'}(\mathbf{s})$. While it is tempting to use this same update rule here, we observe that this update does not provide the meta-learner with sufficient expressive power to be a universal update procedure for the policy, using universality in the sense used by Finn and Levine (2018). For MAML-based methods to approximate any learning procedure, the inner gradient must not discard information needed to infer the task (Finn and Levine, 2018). The gradient of the AWR objective does not contain full information of both the regression weight and the regression target. That is, one cannot recover both the advantage weight and the action from the gradient. We formalize this problem in Theorem 1 in Appendix A. To address this issue and make our meta-learner sufficiently expressive, the MACAW policy update performs both advantage-weighted regression onto actions as well as an additional regression onto action advantages. This enriched policy update is only used during adaptation, and the predicted advantage is used only to enrich the inner loop policy update during meta-training; during meta-test, this predicted advantage is discarded. We prove the universality of the enriched policy update in Theorem 2 in Appendix A. We observe empirically the practical impact of the universality property with an ablation study presented in Figure 5 (left).

To make predictions for both the AWR loss and advantage regression, our policy architecture has two output heads corresponding to the predicted action given the state, $\pi_\theta(\cdot|\mathbf{s})$, and the predicted advantage given both state and action $A_\theta(\mathbf{s},\mathbf{a})$. This architecture is shown in Figure 2. Policy adaptation proceeds as:

$$\theta_i' \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi(\theta, \phi_i', D_i^{\text{tr}}), \text{ where } \mathcal{L}_\pi = \mathcal{L}^{\text{AWR}} + \lambda \mathcal{L}^{\text{ADV}} \tag{3}$$

In our policy update, we show only one gradient step for conciseness of notation, but it can be easily extended to multiple gradient steps. The AWR loss is given in Equation 1, and the advantage regression loss $\mathcal{L}^{\text{ADV}}$ is given by:

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

Adapting with $\mathcal{L}_\pi$ rather than $\mathcal{L}^{\text{AWR}}$ addresses the expressiveness problems noted earlier. This adaptation process is done both in the inner loop of meta-training and during meta-test time, as outlined in Algorithm 2. MACAW is thus *consistent* at meta-test time because it executes a well-defined RL fine-tuning subroutine based on AWR during adaptation. Next, we describe the meta-training procedure for learning the meta-parameters θ and ϕ, the initializations of the policy and value function, respectively.

4.2. Outer-Loop MACAW Procedure

To enable rapid adaptation at meta-test time, we meta-train a set of initial parameters for both the value function and policy to optimize the AWR losses $\mathcal{L}_V$ and $\mathcal{L}^{\text{AWR}}$, respectively, after adaptation (L9-10 in Algorithm 1). We sample a batch of data D_i^{ts} for the outer loop update that is disjoint from the adaptation data D_i^{tr} in order to promote few-shot generalization rather than memorization of the adaptation data. The meta-learning procedure for the value function follows MAML, using the supervised Monte Carlo objective:

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

where $\mathcal{L}_V$ is defined in Equation 2. This objective optimizes for a set of initial value function parameters such that one or a few inner gradient steps lead to an accurate value estimator.

Unlike the inner loop, we optimize the initial policy parameters in the outer loop with a standard advantage-weighted regression objective, since expressiveness concerns only pertain to the inner loop where only a small number of gradient steps are taken. Hence, the meta-objective for our initial policy parameters is as follows:

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

where $\mathcal{L}_\pi$ is defined in Equation 3 and $\mathcal{L}^{\text{AWR}}$ is defined in Equation 1. Note we use the adapted value function for policy adaptation. The complete MACAW algorithm is summarized in Algorithm 1.

4.3. MACAW Architecture

MACAW's enriched policy update (Equation 3) is motivated by the desire to make inner loop policy updates more expressive. In addition to augmenting the objective, we can also take an architectural approach to increasing gradient expressiveness. Recall that for an MLP, a single step of gradient descent can only make a rank-1 update to each weight matrix. Finn and Levine (2018) show that this implies that MLPs must be impractically deep for MAML to be able to produce *any* learning procedure. However, we can shortcut this rank-1 limitation with a relatively simple change to the layers of an MLP, which we call a *weight transform* layer. For a complete mathematical description of this strategy, see Appendix B. This layer maps a latent code into the layer's weight matrix and bias, which are then used to compute the layer's output just as in a typical fully-connected layer. This 'layer-wise linear hypernetwork' (Ha et al., 2016) doesn't change the class of functions computable by the layer on its input, but it increases the expressivity of MAML's gradient. Because we update the latent code by gradient descent in the inner loop (which is mapped back into a new weight matrix and bias in the forward pass) we can, in theory, acquire weight matrix updates of rank up to the dimensionality of the latent code. We use this strategy for all of the weights in both the value function network and the policy network (for all layers in the body and both heas shown in Figure 2). This architecture shares the motivation of enabling more expressive adaptation without increasing model expressiveness described in (Arnold et al., 2021) and is similar to latent embedding optimization (Rusu et al., 2019). However, the choice of using simple linear mapping functions allows us to apply weight transform layers to the entire network while still providing more expressive gradients, and we allow each layer's parameters to be constructed from a different latent code. Our ablation experiments find that this layer improves learning speed and stability.

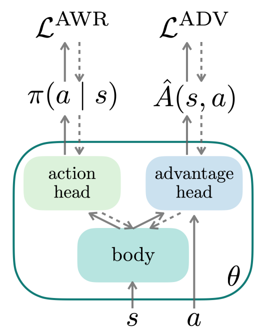

Figure 2: MACAW policy architecture. Solid lines show forward pass; dashed lines show gradient flow during backward pass *during adaptation only*; the advantage head is not used in the outer loop policy update.

5. Experiments

The primary goal of our empirical evaluations is to test whether we can acquire priors from offline multi-task data that facilitate rapid transfer to new tasks. Our evaluation compares MACAW with three sensible approaches to this problem: meta-behavior cloning, multi-task offline RL with fine-tuning, and an offline variant of the state-of-the-art off-policy meta-RL method, PEARL (Rakelly et al., 2019). Further, we evaluate a) MACAW's ability to leverage *online* fine-tuning at meta-test time; b) the importance of MACAW's enriched policy update (Equation 3) and weight transformation (Appendix B); and c) how each method's performance is affected when the sampling of the task space during training is very sparse. To do so, we construct offline variants of the widely-used simulated continuous control benchmark problems introduced by Finn et al. (2017a); Rothfuss et al. (2018), including the half-cheetah with varying goal directions and varying goal velocities, the walker with varying physical parameters, and the ant with varying goal directions. For our main comparison (Figure 3), the offline data for each experiment is generated from the replay buffer of a RL agent trained from scratch. This reflects a practical scenario where an agent has previously learned a

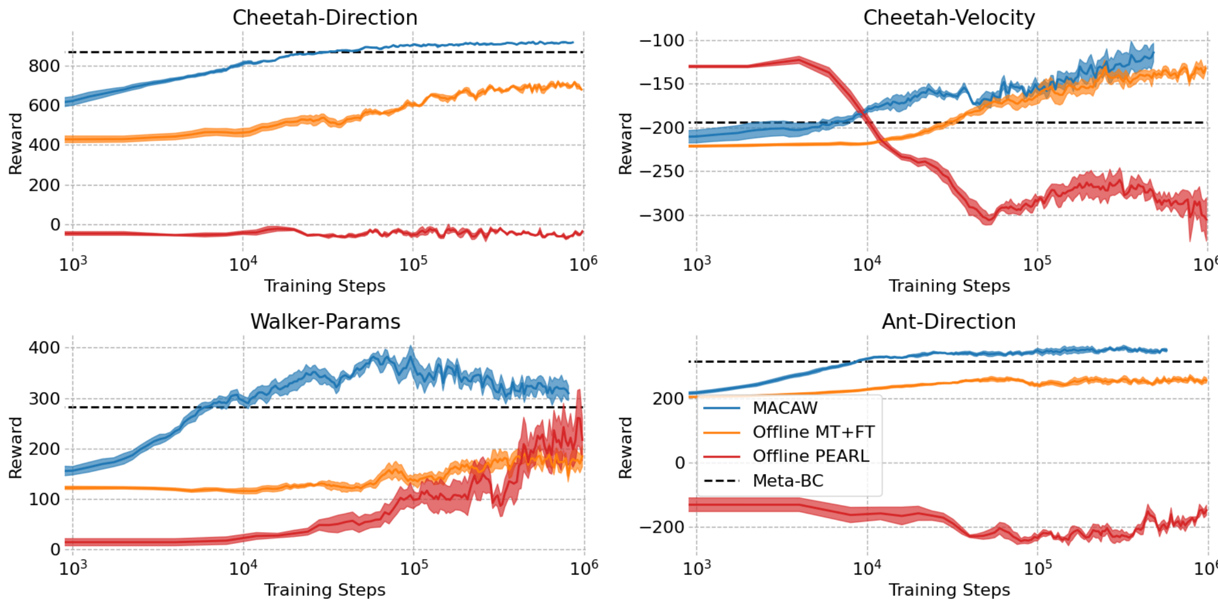

Figure 3: Comparing MACAW with (i) an offline variant of PEARL (Rakelly et al., 2019), a state-of-the-art off-policy meta-RL method, (ii) an offline multi-task training + fine tuning method based on AWR (Peng et al., 2019), and (iii) a meta-behavior cloning baseline. Shaded regions show one standard error of the mean reward of four seeds. MACAW is the only algorithm to consistently outperform the imitation learning baseline, and also learns with the fewest number of training steps in every environment (note the log x axis).

set of tasks via RL, stored its experiences, and now would like to quickly learn a related task. In our ablations, we vary the offline data quantity and quality to better evaluate robustness to these factors. Data collection information is available in Appendix D.

Can we learn to adapt to new tasks quickly from purely offline data? Our first evaluation compares three approaches to the offline meta-RL problem setting, testing their ability to leverage the offline task datasets in order to quickly adapt to a new task. Specifically, we compare MACAW with i) offline PEARL (Rakelly et al., 2019), ii) multi-task AWR (Peng et al., 2019), which uses 20 steps of Adam (Kingma and Ba, 2015) to adapt to a new task at meta-test time (Offline MT+FT) and iii) a meta-behavior cloning baseline. We choose PEARL and AWR because they achieve state-of-the-art performance in off-policy meta-RL and offline RL, respectively, and are readily adaptable to the offline meta-RL problem. As in (Rakelly et al., 2019), for each experiment, we sample a finite set of training tasks and held out test tasks upfront and keep these fixed throughout training. Figure 3 shows the results. We find that MACAW is the only algorithm to consistently outperform the meta-behavior cloning baseline. Multi-task AWR + fine-tuning makes meaningful progress on the simpler cheetah problems, but it is unable to adapt well on the more challenging walker and ant problems. Offline PEARL shows initial progress on cheetah-velocity and walker-params, but struggles to make steady progress on any of the problems. We attribute PEARL's failure to Q-function extrapolation error, a problem known to affect many off-policy RL algorithms (Fujimoto et al., 2019), as well as generally unstable offline bootstrapping. MACAW's and AWR's value function is bootstrap-free and their policy updates maximize a weighted maximum likelihood objective during training, which biases the policy toward safer actions (Peng et al., 2019), implicitly avoiding problems caused by extrapolation error. In contrast to Offline PEARL and multi-task AWR, MACAW trains efficiently and relatively stably on all problems, providing an approach to learning representations from multi-task offline data that can be effectively adapted to new tasks at meta-test time.

Can MACAW leverage online experience at meta-test time? Ideally, an offline meta-RL algorithm should be able to leverage both offline and online data at meta-test time. Here, we evaluate MACAW's and PEARL's ability to improve over offline-only meta-test performance using additional environment interactions collected online in the Cheetah-Velocity, Walker-Params, and Ant-Direction problems. For both algorithms, we perform the initial offline adaptation with a single batch of 256 transitions from μ_{test_i} for each test task $\mathcal{T}_{\text{test}_i}$, as in the fully offline setting. Then, we run online training (RL), alternating between actor/critic updates and online data collection for 10k and 20k new environment steps. The results are reported in Table 1. We find

Problem	MACAW (Ours)			Offline PEARL		
# Online Steps	0	10k	20k	0	10k	20k
Cheetah-Vel	-121.6	-64.0	-60.5	-273.6	-301.8	-297.1
Walker-Params	324.9	286.0	296.9	204.6	117.8	178.3
Ant-Dir	251.9	370.2	376.5	-135.3	57.0	123.9

Table 1: Comparing MACAW and PEARL's average return on held-out test tasks after offline adaptation (0 steps) followed by online fine-tuning with 10k and 20k online interactions. MACAW achieves better performance in all configurations.

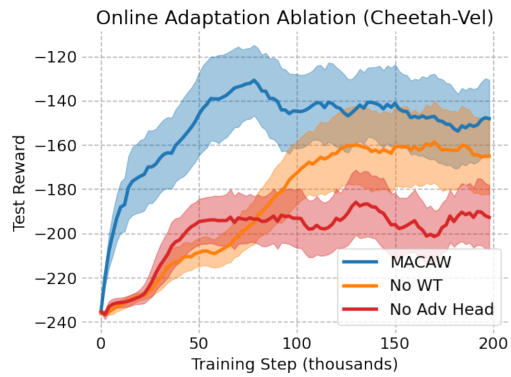

Figure 4: Additional ablation of MACAW using online adaptation data. Adaptation data for both train and test tasks is collected using rollouts of a randomly initialized exploration policy, rather than offline replay buffers of trained RL agents.

that MACAW improves over offline performance with 10k environment steps for 2 out of 3 problems, while PEARL is unable to improve over offline-only performance using either 10k or 20k steps of online experience for 2 out of 3 problems. We attribute MACAW's ability to better leverage online experience to the fact that MACAW is a gradient-based meta-RL algorithm, which explicitly meta-trains for fine-tunability. Although we meta-train with only a single inner-loop gradient step, past work has demonstrated that MAML can continue to improve at meta-test time with more adaptation gradient steps than were using during meta-training (Finn et al., 2017a). Walker-Params proves to be the most difficult environment for both algorithms to leverage online experience, perhaps due to the fact that the environment dynamics change across tasks in this problem, rather than the reward function.

How does MACAW's performance differ from MAML+AWR? MACAW has two key features distinguishing it from MAML+AWR: the enriched policy loss and weight transform layers. Here, we use the Cheetah-Velocity setting to test the effects of both of these components. Ablating the enriched policy loss amounts to optimizing Equation 1 rather than Equation 3 during adaptation; ablating the weight transform layers corresponds to using standard fully-connected layers in both policy and value function. Our first ablation uses a random exploration policy rather than offline data for adaptation, as we hypothesize this setting is most difficult for the naive MAML+AWR algorithm due to the low signal-to-noise ratio in the adaptation data. The results are shown in Figure 4. In this experiment, we use a randomly initialized exploration policy π_{exp} to sample an offline batch (30 trajectories) of adaptation data $\tilde{D}_i$ for each meta-training task. During meta-training, we sample D_i^{tr} from $\tilde{D}_i$ rather than D_i (this corresponds to a minor change to L6 of Algorithm 1). During testing, we use trajectories sampled from π_{exp} rather than the pre-collected offline buffer D to perform adaptation. We find that MACAW provides much faster learning with both the enriched policy loss and weight transform layers than either ablated algorithm. The degradation in asymptotic performance as a result of removing the enriched policy loss is particularly striking; the random exploration data makes disambiguating the task using the standard AWR policy loss difficult, reinforcing the need for a universal policy update procedure in order to adapt to new tasks successfully. Our next ablation experiment more closely examines the effect of the quality of the adaptation data on the post-adaptation policy performance.

To identify when policy update expressiveness is most crucial, we return to the fully offline setting in order to systematically vary the quality of the data sampled during adaptation for both meta-training and meta-testing. We run three variations of this ablation study, using the first 100k, middle 100k, and final 100k transitions in the replay buffer for the offline policies as proxies for poor, medium, and expert offline trajectories. The offline policy learning curves in Appendix D.2 show that this is a reasonable heuristic. We use expert outer loop data (last 100k steps) for all experiments, to ensure that any failures are due to inner loop data quality. Figure 5 (left) shows the results. The ablated algorithm performs well when the offline adaptation data comes from a near-optimal policy, which is essentially a one-shot imitation setting (orange); however, when the offline adaptation data comes from a policy pre-convergence, the difference between MACAW and the ablated algorithm becomes larger (blue and red). This result supports the intuition that policy update expressiveness is of greater importance when the adaptation data is more random, because in this case the adaptation data includes a weaker signal from which to infer the task (e.g. the task cannot be inferred by simply looking at the states visited). Because an agent is unable to collect further experience from the environment during offline adaptation, it is effectively at the mercy of the quality of the behavior policy that produced the data.

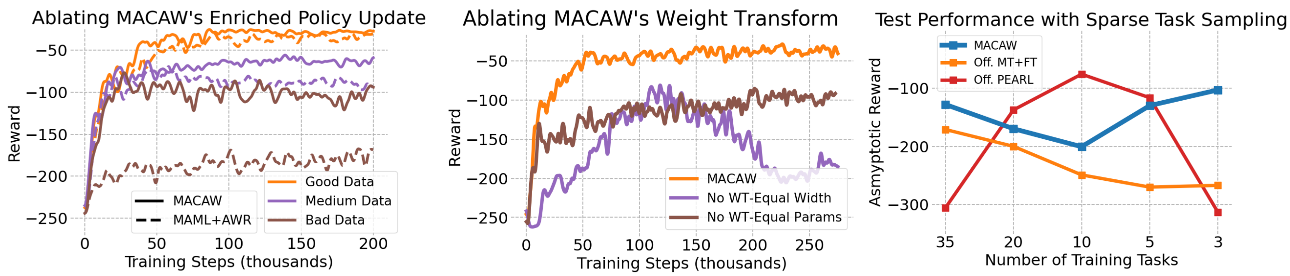

Figure 5: **Left**: Ablating MACAW's enriched policy update when varying the quality of the **inner loop** adaptation data. Solid lines correspond to MACAW, dashed lines correspond to MACAW without the enriched policy update. As data quality worsens, the enriched update becomes increasingly important. Bad, medium, and good data correspond to the first, middle, and last 500 trajectories from the lifetime replay buffer of the behavior policy for each task. **Center**: Ablating MACAW's weight transform layer in the same experimental setting as the cheetah-velocity experiment in Figure 3. Without the additional expressiveness, learning is much slower and less stable. **Right**: Train task sparsity split performance of MACAW, Offline PEARL, and Offline MT+fine tune. MACAW shows the most consistent performance when different numbers of tasks are used, performing well even when only **three tasks** are used for training.

Next, we ablate the weight transform layers, comparing MAML+AWR+enriched policy update with MACAW. Figure 5 (center) suggests that the weight transform layers significantly improve both learning speed and stability. The No WT-Equal Width variant removes the weight transform from each fully-connected layer, replacing it with a regular fully-connected layer of equal width in the forward pass. The No WT-Equal Params variant replaces each of MACAW's weight transform layers with a regular fully-connected layer of greater (constant for all layers) width, to keep the total number of learnable parameters in the network roughly constant. In either case, we find that MACAW provides a significant improvement in learning speed, as well as stability when compared to the Equal Width variant. Figure 6 in the appendix shows that this result is consistent across problems.

How do algorithms perform with varying numbers of meta-training tasks? Generally, we prefer an offline meta-RL algorithm that can generalize to new tasks when presented with only a small number of meta-training tasks sampled from $p(\mathcal{T})$. In this section, we conduct an experiment to evaluate the extent to which various algorithms rely on dense sampling of the space of tasks during training in order to generalize well. We compare the test performance of MACAW, offline PEARL, and offline multi-task AWR + fine-tuning as we hold out an increasingly large percentage of the Cheetah-Velocity task space. Because offline PEARL was unsuccessful in the original Cheetah-Velocity setting in Figure 3, we collect 7x more offline data per task to use during training in this experiment, finding that this allows offline PEARL to learn more successfully, leading to a more interesting comparison. The results are presented in Figure 5 (right). Surprisingly, Offline PEARL completely fails to learn both when training tasks are plentiful and when they are scarce, but learns relatively effectively in the middle regime (5-20 tasks). See Luna Gutierrez and Leonetti (2020) for in-depth discussion of when more training tasks can *hurt* performance in meta-RL. In our experiments, we often observe instability in Offline PEARL's task inference and value function networks when training on too many offline tasks. On the other hand, with too few tasks, the task inference network simply provides insufficient information for the value functions or policy to identify the task. The multi-task learning + fine-tuning baseline exhibits a steadier degradation in performance as training tasks are removed, likely owing to its bootstrap-free learning procedure. Similarly to Offline PEARL, it is not able to learn a useful prior for fine-tuning when only presented with 3 tasks for training. However, MACAW finds a solution of reasonable quality for any sampling of the task space, even for very dense or very sparse samplings of the training tasks. In practice, this property is desirable, because it allows the same algorithm to scale to very large offline datasets while still producing useful adaptation behaviors for small datasets. We attribute MACAW's robustness the strong prior provided by SGD-based adaptation during both meta-training and meta-testing.

6. Related Work

Meta-learning algorithms enable efficient learning of new tasks by learning elements of the learning process itself (Schmidhuber, 1987; Bengio et al., 1992; Thrun and Pratt, 1998; Finn, 2018). We specifically consider the problem of meta-reinforcement learning. Prior methods for meta-RL can generally be categorized into two groups. Contextual meta-RL methods condition a neural network on experience using a recurrent network (Wang et al., 2016; Duan et al., 2016; Fakoor et al., 2020), a recursive network (Mishra et al., 2017), or a stochastic inference network (Rakelly et al., 2019; Zintgraf et al., 2020; Humplik et al., 2019; Sæmundsson et al., 2018). Optimization-based meta-RL methods embed an optimization procedure such as gradient descent into the meta-level optimization (Finn et al., 2017a; Nagabandi et al., 2019; Rothfuss et al., 2018; Zintgraf et al., 2019; Gupta et al., 2018; Mendonca et al., 2019; Yang et al., 2019), potentially using a learned loss function (Houthooft et al., 2018; Bechtle et al., 2019; Kirsch et al., 2020). In prior works, the former class of approaches tend to reach higher asymptotic performance, while the latter class is typically more robust to out-of-distribution tasks, since the meta-test procedure corresponds to a well-formed optimization. Concurrent work by Dorfman and Tamar (2020) investigates the offline meta-RL setting, directly applying an existing meta-RL algorithm, VariBAD (Zintgraf et al., 2020), to the offline setting. The proposed method further assumes knowledge of the reward function for each task to relabel rewards and share data across tasks with shared dynamics. MACAW does not rely on this knowledge nor the assumption that some tasks share dynamics, but this technique could be readily combined with MACAW when these assumptions do hold.

Unlike these prior works, we aim to develop an optimization-based meta-RL algorithm that can both learn from entirely offline data and produces a monotonic learning procedure. Only a handful of previous model-free meta-RL methods leverage off-policy data at all (Rakelly et al., 2019; Mendonca et al., 2019), although one concurrent work does consider the fully offline setting (Dorfman and Tamar, 2020). Guided meta-policy search (Mendonca et al., 2019) is optimization-based but not applicable to the batch setting as it partially relies on policy gradients. Finally, PEARL (Rakelly et al., 2019) and its relatives (Fakoor et al., 2020) correspond to a contextual meta-learning approach sensitive to the meta-training task distribution without fine-tuning (Fakoor et al., 2020) at test time. We also compare to PEARL, and find that, as expected, it performs worse than in the off-policy setting, since the fully offline setting is substantially more challenging than the off-policy setting that it was designed for. This paper builds on the idea of batch off-policy or offline reinforcement learning (Fujimoto et al., 2019; Kumar et al., 2019b; Wu et al., 2019; Levine et al., 2020; Agarwal et al., 2020), extending the problem setting to the meta-learning setting. Several recent works that have successfully applied neural networks to offline RL (Fujimoto et al., 2019; Jaques et al., 2019; Kumar et al., 2019a; Wu et al., 2019; Peng et al., 2019; Agarwal et al., 2020). We specifically choose to build upon the advantage-weighted regression (AWR) algorithm (Peng et al., 2019). We find that AWR performs well without requiring dynamic programming, using Monte Carlo estimation to infer the value function. This property is appealing, as it is difficult to combine truncated optimization-based meta-learners such as MAML (Finn et al., 2017a) with TD learning, which requires a larger number of gradient steps to effectively back-up values.

7. Limitations & Future Work

While MACAW is able to adapt to new tasks from offline data, MACAW does not learn an exploration policy from offline meta-training. An interesting direction for future work is to consider how an agent might learn better *online* exploration policies than the random policies used in our experiments using only offline data during meta-training. The problem of learning to explore has largely been considered in on-policy settings in the past (Gupta et al., 2018; Zintgraf et al., 2020) but also recently in offline settings (Dorfman and Tamar, 2020). In addition, MACAW uses regression onto Monte Carlo returns rather than bootstrapping to fit its value function, which could reduce asymptotic performance during online fine-tuning in some cases. Future work might investigate alternatives such as TD-lambda. Finally, the result of the task sparsity ablation suggests that MACAW learns non-trivial adaptation procedures even with a small number of meta-training tasks; this suggests that MACAW might prove useful in a continual or sequential learning setting, in which adapting quickly after training on only a small number of tasks is particularly valuable.

8. 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 with other state-of-the-art meta-RL algorithms. We motivated the design of MACAW by the desire to build an offline 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 consider fully offline meta-training and meta-testing both with and without online adaptation or fine-tuning, showing that MACAW is effective both when collecting online data is totally infeasible as well as when some online data collection is possible at meta-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 from previous tasks and adapt to new tasks more quickly.

9. Acknowledgements

We acknowledge Saurabh Kumar for helpful discussions. Eric Mitchell is funded by a Knight-Hennessy Fellowship. Xue Bin Peng is funded by a NSERC Postgraduate Scholarship and Berkeley Fellowship for Graduate Study. This work was supported in part by Intel Corporation.

References

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

Sébastien M. R. Arnold, Shariq Iqbal, and Fei Sha. When maml can adapt fast and how to assist when it cannot. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 244–252. PMLR, 13–15 Apr 2021. URL http://proceedings.mlr.press/v130/arnold21a.html.

Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta-learning via learned loss. *arXiv preprint arXiv:1906.05374*, 2019.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, volume 2. Univ. of Texas, 1992.

Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A Meta-Learning library for PyTorch, 2019. URL https://arxiv.org/abs/1909.06576. Available at: https://github.com/tristandeleu/pytorch-meta.

Ron Dorfman and Aviv Tamar. Offline meta reinforcement learning. *arXiv preprint arXiv:2008.02598*, 2020.

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.

Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J. Smola. Meta-q-learning. In *International Conference on Learning Representations*, 2020.

Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *International Conference on Learning Representations*, 10 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017a.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *CoRL*, abs/1709.04905, 2017b. URL http://arxiv.org/abs/1709.04905.

Chelsea B Finn. *Learning to Learn with Gradients*. University of California, Berkeley, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/fujimoto19a.html.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.

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*, 2018.

Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *Advances in Neural Information Processing Systems*, pages 5400–5409, 2018.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May*

7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

Louis Kirsch, Sjoerd van Steenkiste, and Juergen Schmidhuber. Improving generalization in meta reinforcement learning using learned objectives. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1evHerYPr.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019a.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *NeurIPS*, 2019b. URL http://arxiv.org/abs/1906.00949.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, December 2015. doi: 10.1126/science.aab3050. URL https://doi.org/10.1126/science.aab3050.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Ricardo Luna Gutierrez and Matteo Leonetti. Information-theoretic task selection for meta-reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20532–20542. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/ec3183a7f107d1b8dbb90cb3c01ea7d5-Paper.pdf.

Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. In *Advances in Neural Information Processing Systems*, pages 9653–9664, 2019.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *arXiv:1707.03141*, 2017.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2019.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets, 2020.

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, page 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*, pages 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.

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

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJgklhAcK7.

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

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1:2, 1987.

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 3630–3638. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

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.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

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

Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Jie Tan, and Chelsea Finn. Norml: No-reward meta learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 323–331. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.

Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, 2019.