CORRECTING DATA DISTRIBUTION MISMATCH IN OFFLINE META-REINFORCEMENT LEARNING WITH FEW-SHOT ONLINE ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline meta-reinforcement learning (offline meta-RL) extracts knowledge from a given dataset of multiple tasks and achieves fast adaptation to new tasks. Recent offline meta-RL methods typically use task-dependent behavior policies (e.g., training RL agents on each individual task) to collect a multi-task dataset and learn an offline meta-policy. However, these methods always require extra information for fast adaptation, such as offline context for testing tasks or oracle reward functions. Offline meta-RL with few-shot online adaptation remains an open problem. In this paper, we first formally characterize a unique challenge under this setting: data distribution mismatch between offline training and online adaptation. This distribution mismatch may lead to unreliable offline policy evaluation and the regular adaptation methods of online meta-RL will suffer. To address this challenge, we introduce a novel mechanism of data distribution correction, which ensures the consistency between offline and online evaluation by filtering out out-of-distribution episodes in online adaptation. As few-shot out-of-distribution episodes usually have lower returns, we propose a ***G**reedy **C**ontext-based data distribution **C**orrection* approach, called GCC, which greedily infers how to solve new tasks. GCC diversely samples "task hypotheses" from the current posterior belief and selects a greedy hypothesis with the highest return to update the task belief. Our method is the first to provide an effective online adaptation without additional information, and can be combined with off-the-shelf context-based offline meta-training algorithms. Empirical experiments show that GCC achieves state-of-the-art performance on the Meta-World ML1 benchmark compared to baselines with/without offline adaptation.

1 INTRODUCTION

Human intelligence is capable of learning a wide variety of skills from past history and can adapt to new environments by transferring skills with limited experience. Current reinforcement learning (RL) has surpassed human-level performance (Mnih et al., 2015; Silver et al., 2017; Hafner et al., 2019) but requires a vast amount of experience. However, in many real-world applications, RL encounters two major challenges: multi-task efficiency and costly online interactions. In multi-task settings, such as robotic manipulation or locomotion (Yu et al., 2020b), agents are expected to solve new tasks with few-shot adaptation with previously learned knowledge. Moreover, collecting sufficient exploratory interactions is usually expensive or dangerous in robotics (Rafailov et al., 2021), autonomous driving (Yu et al., 2018), and healthcare (Gottesman et al., 2019). One popular paradigm for breaking this practical barrier is *offline meta reinforcement learning* (offline meta-RL; Li et al., 2020; Mitchell et al., 2021), which trains a meta-RL agent with pre-collected offline multi-task datasets and enables fast policy adaptation to unseen tasks.

Recent offline meta-RL methods have been proposed to utilize a multi-task dataset collected by task-dependent behavior policies (Li et al., 2020; Mitchell et al., 2021; Dorfman et al., 2021). They show promise by solving new tasks with few-shot adaptation. However, existing offline meta-RL approaches require additional information or assumptions for fast adaptation. For example, FOCAL (Li et al., 2020) and MACAW (Mitchell et al., 2021) conduct offline adaptation and offline contexts are available for unseen tasks. BORel (Dorfman et al., 2021) and SMAC (Pong et al., 2022) employ few-shot online adaptation, but the former assumes known reward functions, and the latter requires free interactions with the environment without reward supervision. Therefore, achieving effective few-shot online adaptation remains an open problem for offline meta-RL.

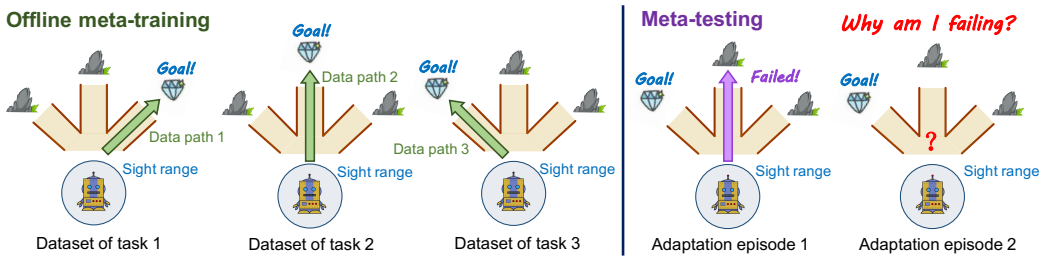


Figure 1: Illustration of data distribution mismatch between offline training and online adaptation.

One particular challenge of meta-RL compared to meta-supervised learning is the need to learn how to explore in the testing environments (Finn & Levine, 2019). In offline meta-RL, the gap of reward and transition distribution between offline training and online adaptation presents a unique conundrum for meta-policy learning, namely ***data distribution mismatch***. As illustrated in Figure 1, when we collect an offline dataset using the expert policy for each task, the robot will be meta-trained on all **successful trajectories** (⟶). The robot will fast explore the environment to reduce task uncertainty in meta-testing. However, when it tries the middle path and hits a stone, this **failed adaptation trajectory** (⟶) does not match the training data distribution, which can lead to false task inference or adaptation. To formally characterize this phenomenon, we utilize the perspective of Bayesian RL (BRL; Duff, 2002; Zintgraf et al., 2019; Dorfman et al., 2021) that maintains a task belief given the context history and learns a meta-policy on belief states. Our theoretical analysis shows that task-dependent data collection results in an inconsistency between offline meta-policy evaluation and online adaptation evaluation, which contrasts with policy evaluation consistency in offline single-task RL (Fujimoto et al., 2019). To deal with this inconsistency, we can choose either to trust the offline dataset or to trust new experience and continue online exploration. The latter may not be able to collect sufficient data in few-shot adaptation, and thus not feasible for meta-RL. Therefore, we adopt the former strategy and introduce a mechanism of online data distribution correction that filters out out-of-distribution episodes and provides a theoretical consistency guarantee in policy evaluations.

To realize our theoretical implications in practical settings, we propose a context-based offline meta-RL algorithm with a novel online adaptation mechanism, called ***G****reedy* ***C****ontext-based data distribution* ***C****orrection* (GCC). To align adaptation context with the meta-training distribution, GCC utilizes greedy task inference, which diversely samples "task hypotheses" and selects a hypothesis with the highest return to update the belief. For example in Figure 1, the robot will try other paths and the failed adaptation trajectory (middle) will not be used for task inference because it is out-of-distribution with a lower return. To our best knowledge, our method is the first to design a delicate context mechanism to achieve effective online adaptation for offline meta-RL and has the advantage of directly combining with off-the-shelf context-based offline meta-training algorithms.

Our main contribution is to formalize a specific challenge (i.e., data distribution mismatch) in offline meta-RL with online adaptation and propose a greedy context mechanism with theoretical motivation. We extensively evaluate the performance of GCC in didactic problems proposed by prior work (Rakelly et al., 2019; Zhang et al., 2021) and Meta-World ML1 benchmark with 50 tasks (Yu et al., 2020b). In these didactic problems, GCC demonstrates that our context mechanism can accurately infer task identification, whereas the original online adaptation methods will suffer due to out-of-distribution data. Empirical results on the more challenging Meta-World ML1 benchmark show that GCC significantly outperforms baselines with few-shot online adaptation, and achieves better or comparable performance than offline adaptation baselines with expert context.

2 Notations and Preliminaries

2.1 Standard meta-RL

The goal of meta-RL (Finn et al., 2017; Rakelly et al., 2019) is to train a meta-policy that can quickly adapt to new tasks using N adaptation episodes. The standard meta-rl setting deals with a distribution $p(\kappa)$ over MDPs, in which each task κ_i sampled from $p(\kappa)$ presents a finite-horizon MDP (Zintgraf et al., 2019; Yin & Wang, 2021). κ_i is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, H, P^{\kappa_i}, R^{\kappa_i})$, including state space $\mathcal{S}$, action space $\mathcal{A}$, reward space $\mathcal{R}$, planning horizon H, transition function $P^{\kappa_i}(s'|s,a)$, and reward function $R^{\kappa_i}(r|s,a)$. Denote $\mathcal{K}$ is the space of task κ_i. In this paper, we assume dynamics function P and reward function R may vary across tasks and share a common structure. The meta-RL

algorithms repeatedly sample batches of tasks to train a meta-policy. In the meta-testing, agents aim to rapidly adapt a good policy for new tasks drawn from $p(\kappa)$.

From a perspective of Bayesian RL (BRL; Ghavamzadeh et al., 2015), recent meta-RL methods (Zintgraf et al., 2019) utilize a Bayes-adaptive Markov decision process (BAMDP; Duff, 2002) to formalize standard meta-RL. A BAMDP is a belief MDP (Kaelbling et al., 1998) of a special Partially Observable MDP (POMDP; Astrom, 1965) whose unobserved state information presents unknown task identification in N adaptation episodes. BAMDPs are defined as a tuple $M^+ = \left(\mathcal{S}^+, \mathcal{A}, \mathcal{R}, H^+, P_0^+, P^+, R^+\right)$ (Zintgraf et al., 2019), in which $\mathcal{S}^+ = \mathcal{S} \times \mathcal{B}$ is the hyper-state space, $\mathcal{B}$ is the space of task beliefs over meta-RL MDPs, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward space, $H^+ = N \times H$ is the planning horizon across adaptation episodes, $P_0^+\left(s_0^+\right)$ is the initial hyper-state distribution representing task distribution $p(\kappa)$, $P^+\left(s_{t+1}^+ \middle| s_t^+, a_t, r_t\right)$ is the transition function, and $R^+\left(r_t \middle| s_t^+, a_t\right)$ is the reward function. A meta-policy $\pi^+\left(a_t \middle| s_t^+\right)$ on BAMDPs prescribes a distribution over actions for each hyper-state $s_t^+ = (s_t, b_t)$. The objective of meta-RL agents is to find a meta-policy π^+ that maximizes expected return, i.e., online policy evaluation denoted by $\mathcal{J}_{M^+}\left(\pi^+\right)$. Formal descriptions are deferred to Appendix A.1.3.

2.2 Offline Meta-RL

In the offline meta-RL setting, a meta-learner only has access to an offline multi-task dataset $\mathcal{D}^+$ and is not allowed to interact with the environment during meta-training (Li et al., 2020). Recent offline meta-RL methods (Dorfman et al., 2021) always utilize task-dependent behavior policies $p(\mu|\kappa)$, which represents the random variable of the behavior policy $\mu(a|s)$ conditioned on the random variable of the task κ. For brevity, we overload $[\mu] = p(\mu|\kappa)$. Similar to related work on offline RL (Yin & Wang, 2021), we assume that $\mathcal{D}^+$ consists of multiple i.i.d. trajectories that are collected by executing task-dependent policies $[\mu]$ in M^+. We Define the reward and transition distribution of the task-dependent data collection by $\mathbb{P}_{M^+,[\mu]}$ (Jin et al., 2021), i.e., for each step t in a trajectory,

$$\mathbb{P}_{M^+,[\mu]}\left(r_t, s_{t+1} \middle| s_t^+, a_t\right) \propto \mathbb{E}_{\kappa_i \sim p(\kappa), \mu_i \sim p(\mu|\kappa_i)}\left[\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \middle| s_t, a_t\right) \cdot p_{M^+}\left(s_t^+ \middle| \kappa_i, \mu_i\right)\right], \quad (1)$$

where the reward and transition distribution of data collection with μ_i in a task κ_i is defined as

$$\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \middle| s_t, a_t\right) = R^{\kappa_i}\left(r_t \middle| s_t, a_t\right) \cdot P^{\kappa_i}\left(s_{t+1} \middle| s_t, a_t\right), \quad (2)$$

and $p_{M^+}\left(s_t^+ \middle| \kappa_i, \mu_i\right)$ denotes the probability of s_t^+ when executing μ_i in a task κ_i. Note that the offline dataset $\mathcal{D}^+$ can be narrow and a large amount of state-action pairs are not covered. These unseen state-action pairs will be erroneously estimated to have unrealistic values, called a phenomenon of *extrapolation error* (Fujimoto et al., 2019). To overcome extrapolation error in offline RL, related works (Fujimoto et al., 2019) introduce batch-constrained RL, which restricts the action space in order to force policy selection of an agent with respect to a given dataset. A policy π^+ is defined to be batch-constrained by $\mathcal{D}^+$ if $\pi^+\left(a \middle| s^+\right) = 0$ for all (s^+, a) tuples that are not contained in $\mathcal{D}^+$. Offline RL (Liu et al., 2020; Chen & Jiang, 2019) approximates policy evaluation for a batch-constrained policy π^+ by sampling from an offline dataset $\mathcal{D}^+$, which is denoted by $\mathcal{J}_{\mathcal{D}^+}\left(\pi^+\right)$ and called *Approximate Dynamic Programming* (ADP; Bertsekas & Tsitsiklis, 1995). During meta-testing, RL agents perform online adaptation using a meta-policy π^+ in new tasks drawn from meta-RL task distribution. The reward and transition distribution of data collection with π^+ in M^+ during adaptation is defined by

$$\mathbb{P}_{M^+}\left(r_t, s_{t+1} \middle| s_t^+, a_t\right) = R^+\left(r_t \middle| s_t^+, a_t\right) \cdot P^+\left(s_{t+1} \middle| s_t^+, a_t\right), \quad (3)$$

where R^+, P^+ are defined in M^+. Detailed formulations are deferred to Appendix A.1.4.

3 Theory: Data Distribution Mismatch Correction

Consistency of training and testing conditions is an important principle for machine learning (Vinyals et al., 2016; Finn & Levine, 2019). Recently, offline meta-RL with task-dependent behavior policies (Li et al., 2020; Mitchell et al., 2021) faces a special challenge: *the reward and transition distribution in offline training and online adaptation may not match*. We first build a theory about data distribution mismatch to understand this phenomenon. Our theoretical analysis is based on Bayesian RL (BRL; Zintgraf et al., 2019) and demonstrates that data distribution mismatch may lead to a large gap between online and offline policy evaluation, a problem which does not appear in offline single-task RL (Fujimoto et al., 2019) settings. To address this challenge, we propose a new mechanism to correct data distribution, which transforms the BAMDPs (Duff, 2002) to enrich information about task-dependent behavior policies into overall beliefs. This transformed BAMDPs provide reliable policy evaluation by requiring the agent to filter out out-of-distribution episodes.

3.1 DATA DISTRIBUTION MISMATCH IN OFFLINE META-RL

We define data distribution mismatch in offline meta-RL as follows.

Definition 1 (Data Distribution Mismatch)**.** *In a BAMDP M^+, for each task-dependent behavior policies $[\mu]$ and batch-constrained meta-policy π^+, the data distribution mismatch between π^+ and $[\mu]$ is defined by that $\exists s_t^+, a_t$, s.t.,*

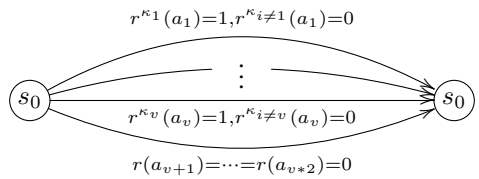
$$p_{M^+}^{\pi^+}\left(s_t^+, a_t\right) > 0 \quad \textit{and} \quad \mathbb{P}_{M^+}\left(r_t, s_{t+1} \middle| s_t^+, a_t\right) \neq \mathbb{P}_{M^+,[\mu]}\left(r_t, s_{t+1} \middle| s_t^+, a_t\right), \tag{4}$$

where $p_{M^+}^{\pi^+}\left(s_t^+, a_t\right)$ is the probability of reaching a tuple $\left(s_t^+, a_t\right)$ while executing π^+ in M^+ (formal definition deferred to Appendix A.2), and $\mathbb{P}_{M^+}, \mathbb{P}_{M^+,[\mu]}$ are the reward and transition distribution of data collection defined in Eq. (1) and (3), respectively.

The data distribution induced by π^+ and $[\mu]$ mismatches when the reward and transition distribution of π^+ and $[\mu]$ differs in a tuple $\left(s_t^+, a_t\right)$, in which the agent can reach this tuple by executing π^+ in M^+, i.e., $p_{M^+}^{\pi^+}\left(s_t^+, a_t\right) > 0$. Note that if π^+ can reach a tuple $\left(s_t^+, a_t\right)$, this tuple is guaranteed to be contained into the offline dataset, i.e., $p_{M^+}^{[\mu]}\left(s_t^+, a_t\right) > 0$, because a batch-constrained policy π^+ will not select an action outside of the dataset collected by $[\mu]$, as introduced in Section 2.2.

Theorem 1. *There exists task-dependent behavior policies $[\mu]$ and a batch-constrained meta-policy π^+ in a BAMDP M^+ such that the data distribution induced by π^+ and $[\mu]$ does not match.*

To serve a concrete example, we construct an offline meta-RL setting shown in Figure 2. In this example, there are v meta-RL tasks $\mathcal{K} = \{\kappa_1, \ldots, \kappa_v\}$ and v behavior policies $\{\mu_1, \ldots, \mu_v\}$, where $v \geq 2$. Each task κ_i has one state s_0, $2v$ actions, and horizon $H = 1$. For each task κ_i, RL agents can receive reward 1 performing action a_i. A common structure shared in $\mathcal{K}$ is that the reward function for actions $\{a_{v+1}, \ldots, a_{2v}\}$ is 0, and the RL agent will not perform these actions to maximize the expected future return. During adaptation, the RL agent can interact with the environment within v epsiodes. The task distribution is uniform, the behavior policy of task κ_i is μ_i, and each behavior policy μ_i will perform a_i. When a batch-constrained meta-policy π^+ selects an action $\tilde{a}$ in the initial state s_0^+, we find that



$p(\kappa_i) = \frac{1}{v}, p(\mu_i|\kappa_i) = 1,$ and $\mu_i(a_i|s_0) = 1$

Figure 2: A concrete example, which has v meta-RL tasks, one state, $2v$ actions, v bahavior polices, horizon $H = 1$ in a episode, and v adaptation episodes, where $v \geq 2$.

$$\mathbb{P}_{M^+}\left(r = 1 \middle| s_0^+, \tilde{a}\right) = \frac{1}{v} \neq \mathbb{P}_{M^+,[\mu]}\left(r = 1 \middle| s_0^+, \tilde{a}\right) = 1, \tag{5}$$

in which there is the probability of $\frac{1}{v}$ to sample a corresponding testing task, whose reward function of $\tilde{a}$ is 1, whereas the reward in the offline dataset collected by $[\mu]$ is all 1.

Proposition 1. *There exists a BAMDP M^+ and a batch-constrained meta-policy π^+ such that, the gap of policy evaluation of π^+ between offline meta-training and online adapation is at least $\frac{H^+}{2}$, where H^+ is the planning horizon of M^+ defined in Section 2.1.*

In the example shown in Figure 2, an offline multi-task dataset D^+ is drawn from the task-dependent data collection $\mathbb{P}_{M^+,[\mu]}$. Since the reward of D^+ is all 1, for each batch-constrained meta-policy π^+, the offline evaluation of π^+ in D^+ is $\mathcal{J}_{D^+}\left(\pi^+\right) = H^+ = vH$, as defined in Section 2.2. The optimal meta-policy $\pi^{+,*}$ in this example is to enumerate $a_1, \ldots, a_v$ until the task identification is inferred from the action with a reward of 1. A meta-policy $\pi^{+,*}$ needs to explore in the testing environments and its online policy evaluation is $\mathcal{J}_{M^+}\left(\pi^{+,*}\right) = \frac{H^+}{2}$. The detailed proof is deferred to Appendix A.2. Thus, the evaluation gap between offline meta-training and online adaptation is

$$\left|\mathcal{J}_{M^+}\left(\pi^+\right) - \mathcal{J}_{D^+}\left(\pi^+\right)\right| \geq \mathcal{J}_{D^+}\left(\pi^+\right) - \mathcal{J}_{M^+}\left(\pi^{+,*}\right) = \frac{H^+}{2}. \tag{6}$$

Proposition 1 suggests that offline policy evaluation may fail and it is imperative to build a connection between online and offline evaluation for effective meta-policy training.

3.2 DATA DISTRIBUTION CORRECTION MECHANISM

To correct data distribution mismatch, we define the transformed BAMDPs as follows.

Definition 2 (Transformed BAMDPs, informal)**.** *A transformed BAMDP is defined by a new BAMDP $\overline{M}^+$, whose belief $\bar{b}_t$ is about meta-RL MDPs κ and task-dependent behavior policies $[\mu]$. The goal of meta-RL agents is to find a meta-policy $\bar{\pi}^+\left(a_t \middle| \bar{s}_t^+\right)$ that maximizes online policy evaluation $\mathcal{J}_{\overline{M}^+}(\bar{\pi}^+)$, where $\bar{s}_t^+ = \left(s_t, \bar{b}_t\right)$ is a hyper-state of $\overline{M}^+$.*

Transformed BAMDPs maintain an overall belief about the task and behavior policies given the current context history. Compared to the original BAMDPs stated in Section 2.2, the transformed BAMDPs incorporate additional information about offline data collection into beliefs. Meanwhile, this transformation introduces an extra condition to the online adaptation process, as indicated by the following fact.

Fact 1. *Transformed BAMDPs implicitly require the agent to filter out out-of-distribution episodes.*

During online adaptation, RL agents construct a hyper-state $\bar{s}_t^+ = \left(s_t, \bar{b}_t\right)$ from the context history and perform a meta-policy $\bar{\pi}^+\left(a_t \middle| \bar{s}_t^+\right)$. The new belief $\bar{b}_t$ accounts for the uncertainty of task MDPs and behavior policies. However, the context history may be conflict with the uncertainty estimation of behavior policies, i.e., RL agents cannot update their beliefs $\bar{b}_t$ when they has observed an event that they believe to have probability zero. For example in Figure 2, RL agents can take an action a_1 and receive a reward of 0 during online adaptation. After obsevering an action a_1, the posterior belief will be $p\left(\kappa_1, \mu_1 \middle| \bar{b}_t\right) = 1$, in which it is contradictory because the reward function of a_1 is 1 in the task κ_1. To support feasible Bayesian belief updating, transformed BAMDPs require the RL agents to filter out out-of-distribution episodes and we can derive the following theorem.

Theorem 2. *In a transformed BAMDP $\overline{M}^+$, for each task-dependent behavior policies $[\mu]$ and batch-constrained meta-policy $\bar{\pi}^+$, the data distribution induced by $\bar{\pi}^+$ and $[\mu]$ matches after filtering out out-of-distribution episodes in online adaptation. Besides, the policy evaluation of $\bar{\pi}^+$ in offline meta-training and online adaptation will be asymptotically consistent, as the offline dataset grows.*

In Theorem 2, the proof of consistent policy evaluation is similar to that in offline single-task RL (Yin & Wang, 2021), whose proof is deferred to Appendix A.2. This theorem indicates that we can design a delicate context mechanism to correct data distribution and guarantee the final performance of online adaptation by maximizing the expected future return during offline meta-training.

4 GCC: GREEDY CONTEXT-BASED DATA DISTRIBUTION CORRECTION

In this section, we investigate a scheme to address data distribution mismatch in offline meta-RL with few-shot online adaptation. Inspired by the theoretical results discussed in Section 3, we aim to distinguish whether an adaptation episode is in the distribution of the given meta-training dataset. RL agents can utilize in-distribution context to infer how to solve meta-testing tasks. However, in practice, accurate out-of-distribution quantification is challenging (Yu et al., 2021; Wang et al., 2021). To address this challenge, we introduce a novel ***G**reedy **C**ontext-based data distribution **C**orrection* (GCC), which elaborates a greedy online adaptation mechanism and can directly combine with off-the-shelf offline meta-RL algorithms. GCC consists of two main components: (i) an off-the-shelf context-based offline meta-training method which extracts meta-knowledge from a given multi-task dataset, and (ii) a greedy context-based online adaptation that realizes a selective context mechanism to infer how to solve meta-testing tasks. The whole algorithm is illustrated in Algorithm 1.

4.1 CONTEXT-BASED OFFLINE META-TRAINING

To support effective offline meta-training, we employ a state-of-the-art off-the-shelf context-based algorithm, i.e., FOCAL (Li et al., 2020), which follows the algorithmic framework of a popular meta-RL approach, PEARL (Rakelly et al., 2019). In this meta-training paradigm, task identification κ is modeled by a latent task variable $z \in \mathcal{Z} = \mathbb{R}^d$, where d is the dimension of the latent space. GCC meta-trains a context encoder $q(z|\boldsymbol{c})$, a policy $\pi(a|s, z)$, and a value function $Q(s, a, z)$, where $\boldsymbol{c}$ is the context information including states, actions, rewards, and next states. The encoder q infers a task belief about the latent task variable z on the space $\mathcal{Z}$ based on the received context. We use $q(z)$ to denote the prior distribution for $\boldsymbol{c} = \emptyset$. The policy π and value function Q are conditioned

on the latent task variable z, in which the representation of z can be end-to-end trained on the RL losses of π or Q. In addition to the gradient from π or Q, recent offline meta-RL (Li et al., 2020) also uses a contrastive loss to help the representation of z distinguish different tasks. We argue that meta-training the inference network q within a given dataset implicitly incorporates the information about offline data collection into the belief, as discussed in Definition 3. Formally, in the offline setting, the prior distribution $q(z)$ can approximately present a distribution of the latent task variable z over each meta-training task, i.e., a sample $\tilde{z}$ drawn from $q(z)$ is equivalent to sampling a task during meta-training and inferring $\tilde{z}$ using the offline context from the given offline dataset. During meta-testing, few-shot out-of-distribution online adaptation episodes may confuse the context encoder $q(z|\boldsymbol{c})$ and RL agents need to use in-distribution context to provide reliable task beliefs.

4.2 Greedy Context-Based Meta-Testing

GCC is a context-based meta-RL algorithm (Rakelly et al., 2019), whose adaptation protocol follows the framework of *posterior sampling* (Strens, 2000). The RL agent will iteratively update task belief by interacting with the environment and improve policy based on belief states. This adaptation paradigm is generalized from Bayesian inference (Thompson, 1933) and has a solid background in RL theory (Agrawal & Goyal, 2012). To ensure a feasible Bayesian belief update, we adopt a common heuristic of offline RL (Fujimoto et al., 2019; Kostrikov et al., 2021), i.e., few-shot out-of-distribution data usually has lower return. This heuristic suggests that the true return of episodes in online adaptation can be a good measure to distinguish data in offline meta-training distribution. In this way, GCC contains two steps to perform a greedy posterior belief update at each iteration: (i) a diverse sampling of latent task variables, and (ii) a greedy context mechanism that selects a task embedding with the highest return to update the belief. For each iteration t, denote the current task belief by b_t, the meta-testing task by κ_{test}, and the number of belief updating iterations by n_{it}.

A diverse sampling of latent task variables will generate n_t^z candidates of the task embedding z_t, denoting by $\mathcal{Z}_t = \left\{z_t^i\right\}_{i=0}^{n_t^z-1}$, to provide various policies $\pi\left(a\,\middle|s, z_t^i\right)$ for the subsequent context selection mechanism. Due to the contrastive loss applied to the representation of z (Li et al., 2020), a closer task embedding z may yield a more similar policy. Hence, to encourage the diversity of policies, each candidate $z_t^i \in \mathcal{Z}_t$ is designed by

$$z_t^i = \arg\max_{\tilde{z}\in\widetilde{\mathcal{Z}}_t^i}\left(\min\left(\min_{j<i}\left\|\tilde{z}-z_t^j\right\|_2, \min_{t'<t,k<n_{t'}^z}\left\|\tilde{z}-z_{t'}^k\right\|_2\right)\right) \text{ and } \widetilde{\mathcal{Z}}_t^i = \left\{\tilde{z}_t^{i,u}\sim b_t\right\}_{u=0}^{n_{\tilde{z}}-1}, \quad (7)$$

in which z_t^i aims to maximize the mimimum distance to the previous embeddings among $n_{\tilde{z}}$ samples. This greedy method is similar to *Farthest-Point Clustering* (Gonzalez, 1985), a two-approximation algorithm for an NP-hard problem *Minimax Facility Location* (Fowler et al., 1981), which seeks a set of locations to minimize the maximum distance from other facilities to the set.

A greedy selective context mechanism will select a latent task variable $z_t \in \mathcal{Z}_t$ with the highest return to update the task belief b_t. To evaluate the return of each z_t^i, GCC utilizes the policy $\pi\left(a\,\middle|s, z_t^i\right)$ to draw n_e episodes in κ_{test}, denoting by $\mathcal{E}_t^i = \left\{e_t^{i,j}\right\}_{j=0}^{n_e-1}$. The policy evaluation of z_t^i can be approximated by sampling:

$$\mathcal{J}_{\kappa_{test}}\left(z_t^i\right) = \mathcal{J}_{\kappa_{test}}\left(\pi\left(a\,\middle|s, z_t^i\right)\right) \approx \widetilde{\mathcal{J}}_{\kappa_{test}}\left(\mathcal{E}_t^i\right) = \frac{1}{n_e}\sum_{j=0}^{n_e-1}\left(\sum_{k=0}^{H-1} r_k\left(e_t^{i,j}\right)\right), \quad (8)$$

where $\widetilde{\mathcal{J}}_{\kappa_{test}}\left(\mathcal{E}_t^i\right)$ is the average return of episodes $\mathcal{E}_t^i$ and $r_k\left(e_t^{i,j}\right)$ is the reward of k-th step in an episode $e_t^{i,j}$. The task belief update in GCC consists of two phases: (i) an initial stage and (ii) an iterative optimization process. In the initial stage, GCC aims to find a reliable initial task inference z_0 using a large amount of n_0^z diverse candidates, i.e., the initial context is $\boldsymbol{c}_0 = \arg\max_{\mathcal{E}_0\in\left\{\mathcal{E}_0^i\right\}}\widetilde{\mathcal{J}}_{\kappa_{test}}\left(\mathcal{E}_0\right)$, maintaining the corresponding task embedding z_0, and deriving the posterior belief $b_1 = q(z|\boldsymbol{c}_0)$. In the following iterations, GCC utilizes an iterative optimization method to maximize final performance during few-shot online adaptation, i.e., when $t > 1$, let $n_t^z = 1$ and if $\widetilde{\mathcal{J}}_{\kappa_{test}}\left(\mathcal{E}_t^0\right) > \widetilde{\mathcal{J}}_{\kappa_{test}}\left(\boldsymbol{c}_{t-1}\right)$, we have $\boldsymbol{c}_t = \mathcal{E}_t^0$ and update the posterior belief $b_{t+1} = q(z|\cup_{t'\leq t}\boldsymbol{c}_{t'})$, otherwise $\boldsymbol{c}_t = \boldsymbol{c}_{t-1}$ and keep the belief $b_{t+1} = b_t$. The final policy $\pi\left(a\,|s, z_t\right)$ will depend on the optimal task embedding z_t with the highest return.

Algorithm 1 GCC: Greedy Context-based data distribution Correction

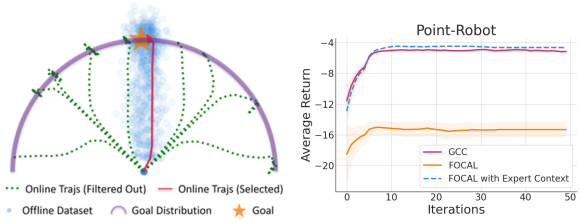
Require: An offline multi-task dataset $\mathcal{D}^+$, a meta-testing task $\kappa_{test} \sim p(\kappa)$, the number of iterations n_{it}, n_0^z, and a context-based offline meta-training algorithm $\mathbb{A}$ (i.e., FOCAL)
Randomly initialize a context encoder $q(z|\boldsymbol{c})$, a policy $\pi(a|s,z)$, and a value function $Q(s,a,z)$
Offline meta-train q, π, and Q given an algorithm $\mathbb{A}$ and a dataset $\mathcal{D}^+$ ▷ *Offline meta-training*
Generate a prior task distribution $q(z)$ using the dataset $\mathcal{D}^+$ ▷ *Start online meta-testing*
Collect diverse adaptation episodes $\{\mathcal{E}_0^i\}_{i=0}^{n_0^z-1}$ using $q(z)$ and π in κ_{test}
Compute the greedy context $\boldsymbol{c}_0$, task embedding z_0, and posterior belief b_1 ▷ *An initial stage*
for $t = 1 \ldots n_{it} - 1$ **do** ▷ *An iterative optimization process*
    Collect a diverse episode $\mathcal{E}_t^0$ using b_t and $\pi(a|s,z_t)$ in κ_{test}
    Compute the greedy context $\boldsymbol{c}_t$ and posterior belief b_{t+1}
Derive the final policy $\pi_{\text{out}}(a|s,z_t)$ with the optimal task embedding z_t
Return: π_{out}

5 EXPERIMENTS

In this section, we first study a didactic example to analyze the out-of-distribution problem, and show how GCC alleviates this problem by its greedy selective mechanism. Then we conduct large-scale experiments on Meta-World ML1(Yu et al., 2020a), a popular meta-RL benchmark that consists of 50 robot arm manipulation task sets. Finally, we perform ablation studies to analyze GCC's sensitivity to hyper-parameter settings and dataset qualities. Following FOCAL (Li et al., 2020), we use expert-level datasets sampled by policies trained with SAC on the corresponding tasks. We compare against FOCAL (Li et al., 2020) and MACAW (Mitchell et al., 2021), as well as their online adaptation variants. FOCAL is built upon PEARL (Rakelly et al., 2019) and uses contrastive losses to learn context embeddings, while MACAW is a MAML-based (Finn et al., 2017) algorithm and incorporates AWR (Peng et al., 2019). Both algorithms are originally proposed for the offline adaptation settings (i.e., with expert context). For online adaptation, we use online experiences instead of expert contexts, and adopt PEARL and MAML's adaptation manner to FOCAL and MACAW, respectively. Results in this section are averaged over six random seeds, and variance is measured by 95% bootstrapped confidence interval. Detailed hyper-parameter settings are deferred to Appendix B.

5.1 DIDACTIC EXAMPLE

We introduce Point-Robot, a simple 2D navigation task set commonly used in meta-RL (Rakelly et al., 2019; Zhang et al., 2021). Figure 3(a) illustrates the distribution mismatch between offline meta-training and online adaptation, as the dataset is collected by task-dependent behavior policies. As a consequence, directly performing adaptation with the online collected trajectories lead to poor adaptation performance, as shown in Figure 3(b). GCC fixes this problem by filtering out out-of-distribution data, and greedily selecting trajectories. At the end of the initial stage, GCC only updates its belief with the orange trajectory as it has the highest return. After the initial stage, GCC iteratively optimizes the posterior belief to get the final policy. As shown in Figure 3(b), GCC achieves comparable performance to FOCAL with expert context and significantly outperforms FOCAL with online adaptation.



(a) (b)

Figure 3: (a) Illustration of data distribution mismatch between offline meta-training (blue) and online adaptation (green and red trajectories). (b) Adaptation performance of GCC, FOCAL, and FOCAL with expert context.

5.2 MAIN RESULTS

We evaluate on Meta-World ML1(Yu et al., 2020a), a popular meta-RL benchmark that consists of 50 robot arm manipulation task sets. Each task set consists of 50 tasks with different goals. For each task set, we use 40 tasks as meta-training tasks, and remain the other 10 tasks as meta-testing tasks. As shown in Table 1, GCC significantly outperforms baselines under the online context setting. With

expert contexts, FOCAL and MACAW both achieves reasonable performance. GCC achieves better or comparable performance to baselines with expert contexts, which implies that expert contexts may not be necessary for offline meta-RL. Under online contexts, FOCAL fails due to the data distribution mismatch between offline training and online adaptation. MACAW has the ability of online fine-tuning as it is based on MAML, but it also suffers from the distribution mismatch problem, and online fine-tuning can hardly improve its performance within a few adaptation episodes.

Table 1: Algorithms' normalized scores averaged over 50 Meta-World ML1 task sets. Scores are normalized by expert-level policy return.

GCC	FOCAL	MACAW	FOCAL with Expert Context	MACAW with Expert Context
0.73 $\pm$ 0.07	0.53 $\pm$ 0.08	0.18 $\pm$ 0.09	0.67 $\pm$ 0.07	0.68 $\pm$ 0.07

Table 2 shows algorithms' performance on 20 representative Meta-World ML1 task sets, as well a sparse-reward version of Point-Robot and Cheetah-Vel, which are popular meta-RL tasks (Li et al., 2020). GCC achieves remarkable performance in most tasks and may fail in some hard tasks as offline meta-training is difficult. We find that GCC achieves better or comparable performance to baselines with expert contexts on 33 out of the 50 task sets. Detailed algorithm performance on all 50 tasks as well as comparison to baselines with expert contexts are deferred to Appendix C.

Table 2: Performance on example tasks, a bunch of Meta-World ML1 tasks with normalized scores.

Example Environments	GCC	FOCAL	MACAW
Coffee-Push-V2	**1.26** $\pm$ 0.13	0.66 $\pm$ 0.07	0.01 $\pm$ 0.01
Faucet-Close-V2	**1.12** $\pm$ 0.01	1.06 $\pm$ 0.02	0.07 $\pm$ 0.01
Faucet-Open-V2	**1.05** $\pm$ 0.02	1.01 $\pm$ 0.02	0.08 $\pm$ 0.04
Door-Close-V2	**0.99** $\pm$ 0.00	0.97 $\pm$ 0.01	0.00 $\pm$ 0.00
Drawer-Close-V2	**0.99** $\pm$ 0.02	**0.96** $\pm$ 0.04	0.53 $\pm$ 0.50
Door-Lock-V2	**0.97** $\pm$ 0.01	0.90 $\pm$ 0.02	0.25 $\pm$ 0.11
Plate-Slide-Back-V2	**0.96** $\pm$ 0.02	0.58 $\pm$ 0.06	0.21 $\pm$ 0.17
Dial-Turn-V2	**0.91** $\pm$ 0.05	0.84 $\pm$ 0.09	0.00 $\pm$ 0.00
Handle-Press-V2	**0.88** $\pm$ 0.05	**0.87** $\pm$ 0.02	0.28 $\pm$ 0.10
Hamme-V2	**0.84** $\pm$ 0.06	0.59 $\pm$ 0.07	0.10 $\pm$ 0.01
Button-Press-V2	**0.74** $\pm$ 0.08	**0.68** $\pm$ 0.14	0.02 $\pm$ 0.01
Push-Wall-V2	**0.71** $\pm$ 0.15	0.43 $\pm$ 0.06	0.23 $\pm$ 0.18
Hand-Insert-V2	**0.63** $\pm$ 0.04	0.29 $\pm$ 0.07	0.02 $\pm$ 0.01
Peg-Unplug-Side-V2	**0.56** $\pm$ 0.07	0.19 $\pm$ 0.09	0.00 $\pm$ 0.00
Bin-Picking-V2	0.53 $\pm$ 0.16	0.31 $\pm$ 0.21	**0.66** $\pm$ 0.11
Soccer-V2	**0.44** $\pm$ 0.04	0.11 $\pm$ 0.03	**0.38** $\pm$ 0.31
Coffee-Pull-V2	**0.40** $\pm$ 0.05	0.23 $\pm$ 0.04	0.19 $\pm$ 0.12
Pick-Place-Wall-V2	0.28 $\pm$ 0.12	0.09 $\pm$ 0.04	**0.39** $\pm$ 0.25
Pick-Out-Of-Hole-V2	0.26 $\pm$ 0.25	0.16 $\pm$ 0.16	**0.59** $\pm$ 0.06
Handle-Pull-Side-V2	**0.14** $\pm$ 0.04	**0.13** $\pm$ 0.09	0.00 $\pm$ 0.00
Cheetah-Vel	**-171.52** $\pm$ 21.96	-287.70 $\pm$ 30.62	-233.97 $\pm$ 23.46
Point-Robot	**-5.10** $\pm$ 0.26	-15.38 $\pm$ 0.95	-14.61 $\pm$ 0.98
Point-Robot-Sparse	**7.78** $\pm$ 0.64	0.83 $\pm$ 0.37	0.00 $\pm$ 0.00

5.3 Ablation Study

In this subsection, we conduct various ablation studies to investigate the robustness of GCC in dataset quality and hyper-parameters.

Initial stage length. Table 3 shows GCC's performance with different initial stage lengths. The total number of adaptation episodes is 20. We find that GCC performs well during 10-15 episodes, which is 50%-75% of the total number of adaptation episodes. A small initial stage length (5) may lead to a possibly unreliable task belief and cause a degrade in performance. The 19-episode does not perform the iterative optimization process, and the task belief updates will not converge.

Number of latent task variables sampled in the initial phase. $n_{\tilde{z}}$ controls the number of diverse samples used to produce the task embedding candidates z_t^i. As shown in Table 4, GCC is robust to changes of $n_{\tilde{z}}$, and works in a large range from 5 to 20.

Dataset Quality. We test GCC and baselines with several "medium" datasets, which are collected by periodically evaluating policies of SAC. As shown in Table 5, GCC still significantly outperforms baseline algorithms on medium datasets, which implies GCC's ability to learn various datasets.

Table 3: GCC's performance with various initial stage lengths.

Environment	5 Episodes	10 Episodes	15 Episodes	19 Episodes
Point-Robot	-6.04 ± 0.31	**-5.11** ± 0.21	**-5.10** ± 0.26	-5.37 ± 0.11
Point-Robot-Sparse	4.04 ± 0.58	**7.78** ± 0.64	**8.07** ± 0.62	7.29 ± 0.50

Table 4: GCC's performance with various $n_{\tilde{z}}$s.

Environment	$n_{\tilde{z}} = 1$	$n_{\tilde{z}} = 5$	$n_{\tilde{z}} = 10$	$n_{\tilde{z}} = 20$
Point-Robot	-5.92 ± 0.31	**-5.11** ± 0.21	**-4.94** ± 0.16	**-4.99** ± 0.23
Point-Robot-Sparse	5.66 ± 0.63	**7.78** ± 0.64	**7.31** ± 0.74	**7.78** ± 0.57

Table 5: Algorithms' performance on datasets of various qualities.

Environment	GCC	FOCAL	MACAW
Sweep-V2	**0.77** ± 0.04	0.32 ± 0.08	0.20 ± 0.20
Sweep-V2-Medium	**0.59** ± 0.13	0.38 ± 0.13	0.04 ± 0.03
Peg-Insert-Side-V2	**0.30** ± 0.04	0.08 ± 0.03	0.00 ± 0.00
Peg-Insert-Side-V2-Medium	**0.30** ± 0.14	0.10 ± 0.07	0.00 ± 0.00

6 Related Work

In the literature, offline meta-RL methods utilize a context-based (Rakelly et al., 2019) or gradient-based (Finn et al., 2017) meta-RL framework to solve new tasks with few-shot adaptation. They utilize the techniques of contrastive learning (Li et al., 2020; Yuan & Lu, 2022), more expressive power (Mitchell et al., 2021), or reward relabeling (Dorfman et al., 2021; Pong et al., 2022) with various popular offline single-task RL tricks, i.e., using KL divergence (Wu et al., 2019; Peng et al., 2019; Nair et al., 2020) or explicitly constraining the policy to be close to the dataset (Fujimoto et al., 2019; Zhou et al., 2020). However, these methods always require extra information for fast adaptation, such as offline context for testing tasks (Li et al., 2020; Mitchell et al., 2021; Yuan & Lu, 2022), oracle reward functions (Dorfman et al., 2021), or free interactions without reward supervision (Pong et al., 2022). To address the challenge, we propose GCC, a greedy context mechanism with theoretical motivation, to perform effective online adaptation without requiring additional information.

The concepts of *distribution shift in z-space* in Pong et al. (2022) and *MDP ambiguity* in Dorfman et al. (2021) are related to the data distribution mismatch proposed in this paper. We reveal that the task-dependent behavior policies will induce different reward and transition distribution between offline meta-training and online adaptation, which is an essential factor in the phenomenon of distribution shift in z-space (Pong et al., 2022). After filtering out these out-of-distribution data, GCC can maintain an overall belief about the task with behavior policies to address MDP ambiguity.

7 Conclusion

This paper formalizes data distribution mismatch in offline meta-RL with online adaptation and introduces GCC, a novel context-based online adaptation approach. Inspired by theoretical implications, GCC adopts a greedy context mechanism to filter out out-of-distribution with lower return for online data correction. We demonstrate that GCC can perform accurate task inference and achieve state-of-the-art performance on Meta-World ML1 benchmark with 50 tasks. Compared to offline adaptation baselines with expert context, GCC also performs better or comparably, suggesting that offline context may not be necessary for the testing environments. One potential future direction is to extend GCC to gradient-based online adaptation methods with data distribution correction.

REFERENCES

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pp. 39–1. JMLR Workshop and Conference Proceedings, 2012.

Alex Slivkins Alekh Agarwal. Lecture 10: Reinforcement learning. In *COMS E6998.001, Columbia University*. 2017. OpenCourseLecture.

Karl J Astrom. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Applic.*, 10:174–205, 1965.

Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE conference on decision and control*, volume 1, pp. 560–564. IEEE, 1995.

Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *Aaai*, volume 94, pp. 1023–1028, 1994.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pp. 1042–1051. PMLR, 2019.

Ron Dorfman, Idan Shenfeld, and Aviv Tamar. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Advances in Neural Information Processing Systems*, 34, 2021.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? *arXiv preprint arXiv:1910.03016*, 2019.

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.

Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

Chelsea Finn and Sergey Levine. Meta-learning: from few-shot learning to rapid reinforcement learning. In *ICML*, 2019.

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

Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137, 1981.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

Omer Gottesman, Fredrik Johansson, Matthieu Komorowski, Aldo Faisal, David Sontag, Finale Doshi-Velez, and Leo Anthony Celi. Guidelines for reinforcement learning in healthcare. *Nature medicine*, 25(1):16–18, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pp. 5084–5096. PMLR, 2021.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

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.

Lanqing Li, Rui Yang, and Dijun Luo. Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *arXiv preprint arXiv:2010.01112*, 2020.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pp. 7780–7791. PMLR, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. In *International Conference on Machine Learning*, pp. 17811–17829. PMLR, 2022.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pp. 1154–1168. PMLR, 2021.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, pp. 5331–5340, 2019.

Tongzheng Ren, Jialian Li, Bo Dai, Simon S Du, and Sujay Sanghavi. Nearly horizon-free offline reinforcement learning. *Advances in neural information processing systems*, 34, 2021.

Laixi Shi, Gen Li, Yuting Wei, Yuxin Chen, and Yuejie Chi. Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity. *arXiv preprint arXiv:2202.13890*, 2022.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

Malcolm Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pp. 943–950, 2000.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Csaba Szepesvári. Lecture 17: Batch rl: Introduction discussion. In *CMPUT 653: Theoretical Foundations of Reinforcement Learning, University of Alberta*. 2022. OpenCourseLecture.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

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

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.

Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.

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

Ming Yin and Yu-Xiang Wang. Towards instance-optimal offline reinforcement learning with pessimism. *Advances in neural information processing systems*, 34, 2021.

Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal provable uniform convergence in offline policy evaluation for reinforcement learning. *arXiv preprint arXiv:2007.03760*, 2020.

Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal offline reinforcement learning via double variance reduction. *Advances in neural information processing systems*, 34, 2021.

Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2(5):6, 2018.

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*, pp. 1094–1100. PMLR, 2020a.

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*, pp. 1094–1100. PMLR, 2020b.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *arXiv preprint arXiv:2102.08363*, 2021.

Haoqi Yuan and Zongqing Lu. Robust task representations for offline meta-reinforcement learning via contrastive learning. In *International Conference on Machine Learning*, pp. 25747–25759. PMLR, 2022.

Jin Zhang, Jianhao Wang, Hao Hu, Tong Chen, Yingfeng Chen, Changjie Fan, and Chongjie Zhang. Metacure: Meta reinforcement learning with empowerment-driven exploration. In *International Conference on Machine Learning*, pp. 12600–12610. PMLR, 2021.

Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. *arXiv preprint arXiv:2011.07213*, 2020.

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*, 2019.

A THEORY

A.1 BACKGROUND

Throughout this paper, for a given non-negative integer $N \in \mathbb{Z}_+$, we use $[N]$ to denote the set $\{0, 1, \ldots, N-1\}$.

A.1.1 FINITE-HORIZON SINGLE-TASK RL

In single-task RL, an agent interacts with a Markov Decision Process (MDP) to maximize its cumulative reward (Sutton & Barto, 2018). A finite-horizon MDP is defined as a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{R}, H, P, R)$ (Zintgraf et al., 2019; Du et al., 2019), where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward space, $H \in \mathbb{Z}_+$ is the planning horizon, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function which takes a state-action pair and returns a distribution over states, and $R : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{R})$ is the reward distribution. In particular, we consider finite state, action, and reward spaces in the theoretical analysis, i.e., $|\mathcal{S}| < \infty, |\mathcal{A}| < \infty, |\mathcal{R}| < \infty$. Without loss of generality, we assume a fixed initial state s_0[1]. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ prescribes a distribution over actions for each state. The policy π induces a (random) H-horizon trajectory $\tau_H^\pi = (s_0, a_0, r_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}, r_{H-1})$, where $a_0 \sim \pi(s_0), r_0 = r(s_0, a_0), s_1 \sim P(s_0, a_0), a_1 \sim \pi(s_1)$, etc. To streamline our analysis, for each $h \in [H]$, we use $\mathcal{S}_h \subseteq \mathcal{S}$ to denote the set of states at h-th timestep, and we assume $\mathcal{S}_h$ do not intersect with each other. To simplify notation, we assume the transition from any state in S_{H-1} and any action to the initial state s_0, i.e., $\forall s \in S_{H-1}, a \in \mathcal{A}$, we have $P(s_0|s, a) = 1$[2]. We also assume $r_t \in [0, 1], \forall t \in [H]$ almost surely. Denote the probability of τ_H:

$$p(\tau_H^\pi) = \left(\prod_{t \in [H]} \pi(a_t|s_t) \cdot R(r_t|s_t, a_t) \right) \prod_{t \in [H-1]} P(s_{t+1}|s_t, a_t). \tag{9}$$

For any policy π, we define a value function $V_\pi : \mathcal{S} \to \mathbb{R}$ as: $\forall h \in [H], \forall s \in \mathcal{S}_h$,

$$V_\pi(s) = \mathbb{E}_{s_h = s, a_t \sim \pi(\cdot|s_t), r_t \sim R(\cdot|s_t, a_t), s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t=h}^{H-1} r_t \right] \tag{10}$$

$$= \begin{cases} \sum_{a \in \mathcal{A}} \pi(a|s) \cdot \mathbb{E}_{r \sim R(\cdot|s,a)}[r], & \text{if } h = H-1, \\ \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathbb{E}_{r \sim R(\cdot|s,a)}[r] + \sum_{s' \in \mathcal{S}_{h+1}} P(s'|s, a) V_\pi(s') \right), & \text{otherwise,} \end{cases} \tag{11}$$

and a visitation distribution of π is defined by $\rho_\pi(\cdot) : \Delta(\mathcal{S})$ which is $\forall h \in [H], \forall s \in \mathcal{S}_h$,

$$\rho_\pi(s) = \begin{cases} \frac{1}{H}, & \text{if } h = 0 \text{ and } s = s_0, \\ \sum_{\tilde{s} \in \mathcal{S}_{h-1}, \tilde{a} \in \mathcal{A}} \rho_\pi(\tilde{s}) \cdot \pi(\tilde{a}|\tilde{s}) \cdot P(s|\tilde{s}, \tilde{a}), & \text{if } h > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

and $\rho_\pi(\cdot, \cdot) : \Delta(\mathcal{S} \times \mathcal{A})$ is $\rho_\pi(s, a) = \rho_\pi(s) \cdot \pi(a|s), \forall s \in \mathcal{S}, a \in \mathcal{A}$. The expected total reward induced by policy π, i.e., the policy evaluation of π, is defined by

$$\mathcal{J}_M(\pi) = V_\pi(s_0) = H \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \rho_\pi(s, a) \cdot \mathbb{E}_{r \sim R(\cdot|s,a)}[r]. \tag{13}$$

The goal of RL is to find a policy π that maximizes its expected return $\mathcal{J}(\pi)$.

[1] Some papers assume the initial state is sampled from a distribution P_1. Note this is equivalent to assuming a fixed initial state s_0, by setting $P(s_0, a) = P_1$ for all $a \in \mathcal{A}$ and now our state s_1 is equivalent to the initial state in their assumption.

[2] The transition from the state in S_{H-1} does not affect learning in the finite-horizon MDP M.

A.1.2 OFFLINE FINITE-HORIZON SINGLE-TASK RL

We consider the offline finite-horizon single-task RL setting, that is, a learner only has access to a dataset $\mathcal{D}$ consisting of K trajectories $\left\{\left(s_t^k, a_t^k, r_t^k\right)\right\}_{t\in[H]}^{k\in[K]}$ (i.e., $|\mathcal{D}| = KH$ tuples) and is not allowed to interact with the environment for additional online explorations. The data can be collected through multi-source logging policies and denote the unknown behavior policy μ. Similar with related work (Ren et al., 2021; Yin et al., 2020; Yin & Wang, 2021; Yin et al., 2021; Shi et al., 2022), we assume that $\mathcal{D}$ is collected through interacting K i.i.d. episodes using policy μ in M. Define the reward and transition distribution of data collection with μ in M by $\mathbb{P}_M$ (Jin et al., 2021), i.e., $\forall t \in [H]$ in each episode,

$$\mathbb{P}_M\left(r_t, s_{t+1} \,|s_t, a_t\right) = R^M\left(r_t \,|s_t, a_t\right) \cdot P^M\left(s_{t+1} \,|s_t, a_t\right), \tag{14}$$

where the action a_t is drawn from a behavior policy μ. Denote a dataset collected following the i.i.d. data collecting process, i.e., $\mathcal{D} \sim (\mathbb{P}_M, \mu)$ is an i.i.d. dataset. Note that the offline dataset $\mathcal{D}$ can be narrow collected by some behavior policy μ and a large amount of state-action pairs are not contained in $\mathcal{D}$. These unseen state-action pairs will be erroneously estimated to have unrealistic values, called a phenomenon *extrapolation error* (Fujimoto et al., 2019). To overcome extrapolation error in policy learning of finite MDPs, Fujimoto et al. (Fujimoto et al., 2019) introduces batch-constrained RL, which restricts the action space in order to force policy selection of an agent with respect to a subset of the given data. Thus, define a batch-constrained policy set is

$$\Pi^{\mathcal{D}} = \left\{\pi \,|\pi(a|s) = 0 \text{ whenever } (s,a) \notin \mathcal{D}\right\}, \tag{15}$$

where denoting $(s,a) \in \mathcal{D}$ if there exists a trajectory containing (s,a) in the dataset $\mathcal{D}$, and similarly for $s \in \mathcal{D}$, $(s,a,r) \in \mathcal{D}$, or $(s,a,r,s') \in \mathcal{D}$. The batch-constrained policy set $\Pi^{\mathcal{D}}$ consists of the policies that for any state s observed in the dataset $\mathcal{D}$, the agent will not select an action outside of the dataset. Thus, for any batch-constrained policy $\pi \in \Pi^{\mathcal{D}}$, define the approximate value function $V_\pi^{\mathcal{D}} : \mathcal{S} \to \mathbb{R}$ estimated from $\mathcal{D}$ (Fujimoto et al., 2019; Liu et al., 2020) as: $\forall h \in [H], \forall s \in \mathcal{S}_h$,

$$V_\pi^{\mathcal{D}}(s) = \mathbb{E}_{s_h=s, a_t\sim\pi(\cdot|s_t), (s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\sum_{t=h}^{H-1} r_t\right] \tag{16}$$

$$= \begin{cases} \sum_{a\in\mathcal{A}} \pi(a|s)\mathbb{E}_{(s,a,r)\in\mathcal{D}}\left[r\right], & \text{if } h = H-1, \\ \sum_{a\in\mathcal{A}} \pi(a|s)\mathbb{E}_{(s,a,r,s')\in\mathcal{D}}\left[r + V_\pi^{\mathcal{D}}(s')\right], & \text{otherwise,} \end{cases} \tag{17}$$

which is called Approximate Dynamic Programming (ADP) (Bertsekas & Tsitsiklis, 1995) and such methods take sampling data as input and approximate the value-function (Liu et al., 2020; Chen & Jiang, 2019). In addition, define the approximate policy evaluation of π estimated from $\mathcal{D}$ as

$$\mathcal{J}_{\mathcal{D}}(\pi) = V_\pi^{\mathcal{D}}(s_0). \tag{18}$$

The offline RL literature (Fujimoto et al., 2019; Liu et al., 2020; Chen & Jiang, 2019; Kumar et al., 2019; 2020) aims to utilize approximate expected total reward $\mathcal{J}_{\mathcal{D}}(\pi)$ with various conservatism regularizations (i.e., policy constraints, policy penalty, uncertainty penalty, etc.) (Levine et al., 2020) to find a good policy within a batch-constrained policy set $\Pi^{\mathcal{D}}$.

A.1.3 STANDARD META-RL

The goal of meta-RL (Finn et al., 2017; Rakelly et al., 2019) is to train a meta-policy that can quickly adapt to new tasks using N adaptation episodes. The standard meta-rl setting deals with a distribution $p(\kappa)$ over MDPs, in which each task κ_i sampled from $p(\kappa)$ presents a finite-horizon MDP (Zintgraf et al., 2019; Du et al., 2019). κ_i is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, H, P^{\kappa_i}, R^{\kappa_i})$, including state space $\mathcal{S}$, action space $\mathcal{A}$, reward space $\mathcal{R}$, planning horizon H, transition function $P^{\kappa_i}(s'|s,a)$, and reward function $R^{\kappa_i}(r|s,a)$. Denote $\mathcal{K}$ is the space of task κ_i. In this paper, we assume dynamics function P and reward function R may vary across tasks and share a common structure. The meta-RL algorithms repeatedly sample batches of tasks to train a meta-policy. In the meta-testing, agents aim to rapidly adapt a good policy for new tasks drawn from $p(\kappa)$.

POMDPs. We can formalize the meta-rl with few-shot adaptation as a specific finite-horizon Partially Observable Markov Decision Process (POMDP), which is defined by a tuple $\widehat{M} = \left(\widehat{\mathcal{S}}, \mathcal{A}, \mathcal{R}, \Omega, \widehat{H}, \widehat{P}, \widehat{P}_0, O, \widehat{R}\right)$, where $\widehat{\mathcal{S}} = \mathcal{S} \times \mathcal{K}$ is the state space, $\mathcal{A}$ and $\mathcal{R}$ are the same action and reward spaces as the finite-horizon MDP M defined in Appendix A.1.1, respectively, $\Omega = \mathcal{S}$ is the observation space, $\widehat{H} = N \times H$ is the planning horizon which represents N adapation episodes for a single meta-rl MDP κ_i, as discussed in Zintgraf et al. (Zintgraf et al., 2019), $\widehat{P} : \widehat{\mathcal{S}} \times \mathcal{A} \to \Delta\left(\widehat{\mathcal{S}}\right)$ is the transition function: $\forall \hat{s}, \hat{s}' \in \widehat{\mathcal{S}}, a \in \mathcal{A}$, where denoting $\hat{s} = (s, \kappa_i)$ and $\hat{s}' = (s', \kappa_j)$,

$$\widehat{P}\left(\hat{s}'|\hat{s}, a\right) = \begin{cases} P^{\kappa_i}(s'|s, a), & \text{if } \kappa_i = \kappa_j, \\ 0, & \text{otherwise,} \end{cases} \tag{19}$$

$\widehat{P}_0 : \Delta\left(\widehat{\mathcal{S}}\right)$ is the initial state distribution: $\forall \hat{s} = (s, \kappa_i) \in \widehat{\mathcal{S}}$,

$$\widehat{P}_0\left(\hat{s}\right) = \begin{cases} p(\kappa_i), & \text{if } s = s_0, \\ 0, & \text{otherwise,} \end{cases} \tag{20}$$

$O : \widehat{\mathcal{S}} \to \Delta\left(\Omega\right)$ is the observation probability distribution conditioned on a state: $\forall \hat{s} = (s, \kappa_i) \in \widehat{\mathcal{S}}, o \in \Omega$,

$$O\left(o|\hat{s}\right) = \begin{cases} 1, & \text{if } o = s, \\ 0, & \text{otherwise,} \end{cases} \tag{21}$$

and $\widehat{R} : \widehat{\mathcal{S}} \times \mathcal{A} \to \Delta\left(\mathcal{R}\right)$ is the reward distribution: $\forall \hat{s} = (s, \kappa_i) \in \widehat{\mathcal{S}}, a \in \mathcal{A}, r \in \mathcal{R}$,

$$\widehat{R}\left(r|\hat{s}, a\right) = R^{\kappa_i}(r|s, a). \tag{22}$$

Denote context $c_t = (a_t, r_t, s_{t+1})$ as an experience collected at timestep t, and $c_{:t} = \langle s_0, c_0, \ldots, c_{t-1}\rangle$[3]$\in \mathcal{C}_t \equiv \Omega \times \left(\mathcal{A} \times \mathcal{R} \times \Omega\right)^t$ indicates all experiences collected during t timesteps. Note that t may be larger than H, and when it is the case, $c_{:t}$ represents experiences collected across episodes in the single meta-rl MDP κ_i. Denote the entire context space $\mathcal{C} = \bigcup_{t=0}^{\widehat{H}-1} \mathcal{C}_t$ and a meta-policy $\hat{\pi} : \mathcal{C} \to \Delta\left(\mathcal{A}\right)$ (Wang et al., 2016; Duan et al., 2016) prescribes a distribution over actions for each context. The goal of meta-rl is to find a meta-policy on history contexts $\hat{\pi}$ that maximizes the expected return within N adaptation episodes:

$$\mathcal{J}_{\widehat{M}}(\hat{\pi}) = \mathbb{E}_{\hat{s}_0 \sim P_0, o_t \sim O(\cdot|s_t), a_t \sim \hat{\pi}(\cdot|c_{:t}), r_t \sim \widehat{R}(\cdot|s_t, a_t), \hat{s}_{t+1} \sim \widehat{P}(\cdot|\hat{s}_t, a_t)} \left[\sum_{t=0}^{\widehat{H}-1} r_t\right] \tag{23}$$

$$= \mathbb{E}_{\kappa_i \sim p(\kappa)} \left[\sum_{j=0}^{N-1} \mathbb{E}_{a_t \sim \hat{\pi}\left(\cdot\middle|c_{:(jH+t)}\right), r_t \sim R^{\kappa_i}(\cdot|s_t, a_t), s_{t+1} \sim P^{\kappa_i}(\cdot|s_t, a_t)} \left[\sum_{t=0}^{H-1} r_t\right]\right]. \tag{24}$$

BAMDPs. A Markovian belief state allows a POMDP to be formulated as a Markov decision process where every belief is a state (Cassandra et al., 1994). We can transform the finite-horizon POMDP $\widehat{M}$ to a finite-horizon belief MDP, which is called Bayes-Adaptive MDP (BAMDP) in the literature (Zintgraf et al., 2019; Ghavamzadeh et al., 2015; Dorfman et al., 2021) and is defined by a tuple $M^+ = \left(\mathcal{S}^+, \mathcal{A}, \mathcal{R}, H^+, P^+, P_0^+, R^+\right)$, $\mathcal{S}^+ = \mathcal{S} \times \mathcal{B}$ is the hyper-state space, where $\mathcal{B} = \left\{p(\kappa|c)\,|c \in \mathcal{C}\right\}$ is the set of belief states over the meta-rl MDPs, the prior

$$b_0^\kappa = p\left(\kappa|c_{:0}\right) = p(\kappa) \tag{25}$$

is the meta-rl MDP distribution, and $\forall t \in \left[\widehat{H} - 1\right], \forall c_{:(t+1)} \in \mathcal{C}$, denoting $b_t^\kappa = p\left(\kappa|c_{:t}\right)$ and

$$b_{t+1}^\kappa = p\left(\kappa|c_{:(t+1)}\right) = p\left(p\left(\kappa|c_{:t}\right)|c_{:(t+1)}\right) = p\left(p\left(\kappa|c_{:t}\right)|s_t, c_t\right) = p\left(b_t^\kappa|s_t, c_t\right) \tag{26}$$

$$\propto p\left(b_t^\kappa, c_t|s_t\right) = p\left(c_t|s_t, b_t^\kappa\right) p\left(b_t^\kappa|s_t\right) = p\left(c_t|s_t, b_t^\kappa\right) b_t^\kappa \tag{27}$$

[3]For clarity, we denote $c_{:0}^{\kappa_i} = s_0$.

$$= \mathbb{E}_{\kappa_i \sim b_t^\kappa} \left[R^{\kappa_i}(r_t|s_t, a_t) \cdot P^{\kappa_i}(s_{t+1}|s_t, a_t) \right] \cdot b_t^\kappa \tag{28}$$

is the posterior over the MDPs given the context $c_{:(t+1)}$, $\mathcal{A}, \mathcal{R}$ are the same action space and reward space as the finite-horizon POMDP $\widehat{M}$, respectively, $H^+ = N \times H$ is the planning horizon across adaptation episodes, $P^+ : \mathcal{S}^+ \times \mathcal{A} \times \mathcal{R} \to \Delta(\mathcal{S}^+)$ is the transition function: $\forall s_t^+, s_{t+1}^+ \in \mathcal{S}^+, a_t \in \mathcal{A}, r_t \in \mathcal{R}$, where denoting $s_t^+ = (s_t, b_t^\kappa)$ and $s_{t+1}^+ = \left(s_{t+1}, \tilde{b}_{t+1}^\kappa\right)$,

$$P^+\left(s_{t+1}^+ \middle| s_t^+, a_t, r_t\right) = P^+\left(s_{t+1}, \tilde{b}_{t+1}^\kappa \middle| s_t, b_t^\kappa, a_t, r_t\right) \tag{29}$$

$$= P^+\left(s_{t+1} \middle| s_t, b_t^\kappa, a_t, r_t\right) P^+\left(\tilde{b}_{t+1}^\kappa \middle| s_t, b_t^\kappa, c_t\right) \tag{30}$$

$$= \mathbb{E}_{\kappa_i \sim p(b_t^\kappa|s_t, a_t, r_t)} \left[P^{\kappa_i}(s_{t+1}|s_t, a_t) \right] \cdot \mathbb{1}\left[\tilde{b}_{t+1}^\kappa = p(b_t^\kappa|s_t, c_t)\right], \tag{31}$$

$P_0^+ : \Delta(\mathcal{S}^+)$ is the initial hyper-state distribution, i.e., a deterministic initial hyper-state is

$$s_0^+ = (s_0, b_0^\kappa) = (s_0, p(\kappa)) \in \mathcal{S}^+, \tag{32}$$

and $R^+ : \mathcal{S}^+ \times \mathcal{A} \to \Delta(\mathcal{R})$ is the reward distribution: $\forall s^+ = (s, b^\kappa) \in \mathcal{S}^+, a \in \mathcal{A}, r \in \mathcal{R}$,

$$R^+\left(r|s^+, a\right) = R^+\left(r|s, b^\kappa, a\right) = \mathbb{E}_{\kappa_i \sim b^\kappa}\left[R^{\kappa_i}(r|s, a)\right]. \tag{33}$$

In a BAMDP, the belief is over the transition and reward functions, which are constant for a given task. A meta-policy on BAMDP $\pi^+ : \mathcal{S}^+ \to \Delta(\mathcal{A})$ prescribes a distribution over actions for each hyper-state. The agent's objective is now to find a meta-policy on hyper-states π^+ that maximizes the expected return in the BAMDP,

$$\mathcal{J}_{M^+}\left(\pi^+\right) = \mathbb{E}_{a_t \sim \pi^+\left(\cdot|s_t^+\right), r_t \sim R^+\left(\cdot|s_t^+, a_t\right), s_{t+1}^+ \sim P^+\left(\cdot|s_t^+, a_t\right)} \left[\sum_{t=0}^{\widehat{H}-1} r_t\right] \tag{34}$$

$$= \mathbb{E}_{\kappa_i \sim p(\kappa)} \left[\sum_{j=0}^{N-1} \mathbb{E}_{a_t \sim \pi^+\left(\cdot|s_{jH+t}^+\right), r_t \sim R^{\kappa_i}(\cdot|s_t, a_t), s_{t+1} \sim P^{\kappa_i}(\cdot|s_t, a_t)} \left[\sum_{t=0}^{H-1} r_t\right]\right]. \tag{35}$$

For any meta-policy on hyper-states π^+, denote the corresponding meta-policy on history contexts $\hat{f}_{\pi^+} : \mathcal{C} \to \Delta(\mathcal{A})$, i.e., $\forall t \in \left[\widehat{H} - 1\right], \forall c_{:t} \in \mathcal{C}_t$, s.t., $\hat{f}_{\pi^+}(\cdot|c_{:t}) = \pi^+\left(\cdot|s_t^+\right)$, where $s_t^+ = (s_t, b_t^\kappa) = (s_t, p(\kappa|c_{:t}))$, and we have

$$\mathcal{J}_{\widehat{M}}\left(\hat{f}_{\pi^+}\right) = \mathbb{E}_{\kappa_i \sim p(\kappa)} \left[\sum_{j=0}^{N-1} \mathbb{E}_{a_t \sim \hat{f}_{\pi^+}\left(\cdot|c_{:(jH+t)}\right), r_t \sim R^{\kappa_i}(\cdot|s_t, a_t), s_{t+1} \sim P^{\kappa_i}(\cdot|s_t, a_t)} \left[\sum_{t=0}^{H-1} r_t\right]\right] \tag{36}$$

$$= \mathcal{J}_{M^+}\left(\pi^+\right). \tag{37}$$

The belief MDP is such that an optimal policy for it, coupled with the correct state estimator, will give rise to optimal behavior for the original POMDP (Astrom, 1965; Smallwood & Sondik, 1973; Kaelbling et al., 1998), which indicates that

$$\mathcal{J}_{M^+}\left(\pi^{+,*}\right) = \mathcal{J}_{\widehat{M}}\left(\hat{f}_{\pi^{+,*}}\right) = \mathcal{J}_{\widehat{M}}\left(\hat{\pi}^*\right), \tag{38}$$

where $\pi^{+,*}$ and $\hat{\pi}^*$ are the optimal policies for BAMDP M^+ and POMDP $\widehat{M}$, respectively. Thus, the agent can find a policy π^+ to maximize the expected return in the BAMDP M^+ to address the POMDP $\widehat{M}$ by the transformed policy $\hat{f}_{\pi^+}$.

A.1.4 Offline meta-RL

In the offline meta-RL setting, a meta-learner only has access to an offline multi-task dataset $\mathcal{D}^+$ and is not allowed to interact with the environment during meta-training (Li et al., 2020). Recent offline meta-RL methods (Dorfman et al., 2021) always utilize task-dependent behavior policies $p(\mu|\kappa)$, which represents the random variable of the behavior policy $\mu(a|s)$ conditioned on the random variable of the task κ. For brevity, we overload $[\mu] = p(\mu|\kappa)$. Similar to related work

on offline RL (Shi et al., 2022), we assume that $\mathcal{D}^+$ is collected through interacting multiple i.i.d. trajectories using task-dependent policies $[\mu]$ in M^+. Define the reward and transition distribution of the task-dependent data collection by $\mathbb{P}_{M^+,[\mu]}$ (Jin et al., 2021), i.e., for each step t in a trajectory,

$$\mathbb{P}_{M^+,[\mu]}\left(r_t, s_{t+1} \left| s_t^+, a_t\right.\right) \propto \mathbb{E}_{\kappa_i \sim p(\kappa), \mu_i \sim p(\mu|\kappa_i)}\left[\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \left| s_t, a_t\right.\right) \cdot p_{M^+}\left(s_t^+ \left| \kappa_i, \mu_i\right.\right)\right], \quad (39)$$

where $\mathbb{P}_{\kappa_i}$ is the reward and transition distribution of κ_i defined in Eq. (14), and $p_{M^+}\left(s_t^+ \left| \kappa_i, \mu_i\right.\right)$ denotes the probability of s_t^+ when executing μ_i in a task κ_i, i.e.,

$$p_{M^+}\left(s_t^+ \left| \kappa_i, \mu_i\right.\right) = \sum_{c_{:t} \in \mathcal{C}_t} p_{\kappa_i}^{\mu_i}(c_{:t}) \cdot \mathbb{1}\left[b_t^\kappa = p(\kappa | c_{:t})\right], \quad (40)$$

where the state in $c_{:t}$ is s_t and $p_{\kappa_i}^{\mu_i}$ is defined in Eq. (9). Similar with offline single-task RL (see Appendix A.1.2), offline dataset $\mathcal{D}^+$ can be narrow and a large amount of state-action pairs are not contained. These unseen state-action pairs will be erroneously estimated to have unrealistic values, called a phenomenon *extrapolation error* (Fujimoto et al., 2019). To overcome extrapolation error in offline RL, related works (Fujimoto et al., 2019) introduce batch-constrained RL, which restricts the action space in order to force policy selection of an agent with respect to a given dataset. Define a policy π^+ to be batch-constrained by $\mathcal{D}^+$ if $\pi^+\left(a \left| s^+\right.\right) = 0$ whenever a tuple (s^+, a) is not contained in $\mathcal{D}^+$. Offline RL (Liu et al., 2020; Chen & Jiang, 2019) approximates policy evaluation for a batch-constrained policy π^+ by sampling from an offline dataset $\mathcal{D}^+$, which is denoted by $\mathcal{J}_{\mathcal{D}^+}\left(\pi^+\right)$ and called *Approximate Dynamic Programming* (ADP; Bertsekas & Tsitsiklis, 1995). During meta-testing, RL agents perform online adaptation using a meta-policy π^+ in new tasks drawn from meta-RL task distribution. The reward and transition distribution of data collection with π^+ in M^+ during adaptation is defined by

$$\mathbb{P}_{M^+}\left(r_t, s_{t+1} \left| s_t^+, a_t\right.\right) = R^+\left(r_t \left| s_t^+, a_t\right.\right) \cdot P^+\left(s_{t+1} \left| s_t^+, a_t\right.\right), \quad (41)$$

where $P^+\left(s_{t+1} \left| s_t^+, a_t\right.\right)$ is the marginal transition functions of M^+, i.e.,

$$P^+\left(s_{t+1} \left| s_t^+, a_t\right.\right) = \mathbb{E}_{\kappa_i \sim b_t^\kappa}\left[P^{\kappa_i}\left(s_{t+1} \left| s_t, a_t\right.\right)\right]. \quad (42)$$

A.2 Main Results

Definition 1 (Data Distribution Mismatch)**.** *In a BAMDP M^+, for each task-dependent behavior policies $[\mu]$ and batch-constrained meta-policy π^+, the data distribution mismatch between π^+ and $[\mu]$ is defined by that $\exists s_t^+, a_t$, s.t.,*

$$p_{M^+}^{\pi^+}\left(s_t^+, a_t\right) > 0 \quad \textit{and} \quad \mathbb{P}_{M^+}\left(r_t, s_{t+1} \left| s_t^+, a_t\right.\right) \neq \mathbb{P}_{M^+,[\mu]}\left(r_t, s_{t+1} \left| s_t^+, a_t\right.\right), \quad (4)$$

where $p_{M^+}^{\pi^+}\left(s_t^+, a_t\right)$ is the probability of reaching a tuple $\left(s_t^+, a_t\right)$ while executing π^+ in M^+ (formal definition deferred to Appendix A.2), and $\mathbb{P}_{M^+}, \mathbb{P}_{M^+,[\mu]}$ are the reward and transition distribution of data collection defined in Eq. (1) and (3), respectively.

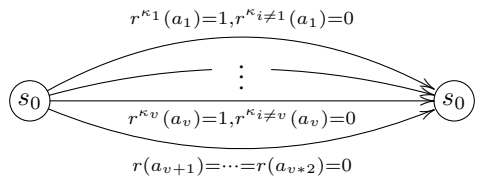
More specifically,

$$p_{M^+}^{\pi^+}\left(s_t^+, a_t\right) = p_{M^+}^{\pi^+}(s_t^+) \cdot \pi^+\left(a_t \left| s_t^+\right.\right), \quad (43)$$

where $p_{M^+}^{\pi^+}(s_t^+)$ is defined in Eq. (9).

Theorem 1. *There exists task-dependent behavior policies $[\mu]$ and a batch-constrained meta-policy π^+ in a BAMDP M^+ such that the data distribution induced by π^+ and $[\mu]$ does not match.*

Proof. To serve a concrete example, we construct an offline meta-RL setting shown in Figure 4. In this example, there are v meta-RL tasks $\mathcal{K} = \{\kappa_1, \ldots, \kappa_v\}$ and v behavior policies $\{\mu_1, \ldots, \mu_v\}$, where $v \geq 2$. Each task κ_i has one state $\mathcal{S} = \{s_0\}$, $2v$ actions $\mathcal{A} = \{a_1, \ldots, a_{2v}\}$, and horizon in an episode $H = 1$. For each task κ_i, RL agents can receive reward 1 performing action a_i. A common structure shared in $\mathcal{K}$ is that the reward function for actions $\{a_{v+1}, \ldots, a_{2v}\}$ is 0, and the RL agent will not perform these actions to maximize the expected future return. During adaptation, the RL agent can interact with the environment within v epsiodes. The task distribution is uniform, the behavior



$$p(\kappa_i) = \frac{1}{v}, p(\mu_i|\kappa_i) = 1, \text{and } \mu_i(a_i|s_0) = 1$$

Figure 4: A concrete example, which has v meta-RL tasks, one state, $2v$ actions, v bahavior polices, horizon $H = 1$ in a episode, and v adaptation episodes, where $v \geq 2$.

policy of task κ_i is μ_i, and each behavior policy μ_i will perform a_i. When a batch-constrained meta-policy π^+ selects an action $\tilde{a}$ in the initial state s_0^+, we find that

$$\mathbb{P}_{M^+}\left(r = 1 \left| s_0^+, \tilde{a}\right.\right) = \frac{1}{v} \neq \mathbb{P}_{M^+,[\mu]}\left(r = 1 \left| s_0^+, \tilde{a}\right.\right) = 1, \tag{44}$$

in which there is the probability of $\frac{1}{v}$ to sample a corresponding testing task, whose reward function of $\tilde{a}$ is 1, whereas the reward in the offline dataset collected by $[\mu]$ is all 1. □

Proposition 1. *There exists a BAMDP M^+ and a batch-constrained meta-policy π^+ such that, the gap of policy evaluation of π^+ between offline meta-training and online adapation is at least $\frac{H^+}{2}$, where H^+ is the planning horizon of M^+ defined in Section 2.1.*

Proof. In the example shown in Figure 4, an offline multi-task dataset D^+ is drawn from the task-dependent data collection $\mathbb{P}_{M^+,[\mu]}$. Since the reward of D^+ is all 1, for each batch-constrained meta-policy π^+, the offline evaluation of π^+ in D^+ is $\mathcal{J}_{D^+}(\pi^+) = H^+ = vH$. The optimal meta-policy $\pi^{+,*}$ in this example is to enumerate $a_1, \ldots, a_v$ until the task identification is inferred from an action with a reward of 1. A meta-policy $\pi^{+,*}$ needs to explore in the testing environments and its online policy evaluation is

$$\mathcal{J}_{M^+}\left(\pi^{+,*}\right) = \sum_{k=0}^{N-1} \frac{N-k}{v-k} \prod_{j=0}^{k-1} \left(1 - \frac{1}{v-j}\right) \tag{45}$$

$$= \sum_{k=0}^{N-1} \frac{v-k}{v} = N - \frac{N^2}{2v} = \frac{v}{2} = \frac{H^+}{2}, \tag{46}$$

where $N = v$ is the number of adaptation episodes. Thus, the gap of policy evaluation of π^+ between offline meta-training and online adapation is

$$\left|\mathcal{J}_{M^+}\left(\pi^+\right) - \mathcal{J}_{D^+}\left(\pi^+\right)\right| \geq \mathcal{J}_{D^+}\left(\pi^+\right) - \mathcal{J}_{M^+}\left(\pi^{+,*}\right) = \frac{H^+}{2}. \tag{47}$$

□

Definition 3 (Transformed BAMDPs (formal version of Definition 2))**.** *A transformed BAMDP is defined as a tuple $\overline{M}^+ = \left(\overline{\mathcal{S}}^+, \mathcal{A}, \mathcal{R}, H^+, \overline{P}^+, \overline{P}_0^+, \overline{R}^+\right)$, where $\overline{\mathcal{S}}^+ = \mathcal{S} \times \overline{\mathcal{B}}$ is the hyper-state space, $\overline{\mathcal{B}}$ is the space of overall beliefs over meta-RL MDPs with behavior policies, $\mathcal{A}, \mathcal{R}, H^+$ are the same action space, reward space, and planning horizon as the original BAMDP M^+, respectively, $\overline{P}_0^+$ is the initial hyper-state distribution presenting joint distribution of task and behavior policies $p(\kappa, \mu) = p(\kappa)p(\mu|\kappa)$, and $\overline{P}^+, \overline{R}^+$ are the transition and reward functions. The goal of meta-RL agents is to find a meta-policy $\bar{\pi}^+\left(a_t \left| \bar{s}_t^+\right.\right)$ that maximizes online policy evaluation $\mathcal{J}_{\overline{M}^+}(\bar{\pi}^+)$.*

More specifically, a finite-horizon transformed BAMDP is defined by a tuple $\overline{M}^+ = \left(\overline{\mathcal{S}}^+, \mathcal{A}, \mathcal{R}, H^+, \overline{P}^+, \overline{P}_0^+, \overline{R}^+\right)$, $\overline{\mathcal{S}}^+ = \mathcal{S} \times \overline{\mathcal{B}}$ is the hyper-state space, where $\overline{\mathcal{B}} = \{p(\kappa, \mu|c) \,|c \in \mathcal{C}\}$ is the set of belief states over the policy-constrained meta-rl MDPs, the prior

$$b_0^{\kappa,\mu} = p\left(\kappa, \mu \,|c_{:0}\right) = p(\kappa, \mu) \tag{48}$$

is the distribution of policy-constrained meta-rl MDPs, and $\forall t \in \left[\widehat{H}-1\right]$, $\forall c_{:(t+1)} \in \mathcal{C}$, denoting $b_t^{\kappa,\mu} = p\left(\kappa,\mu|c_{:t}\right)$ and

$$b_{t+1}^{\kappa,\mu} = p\left(\kappa,\mu\left|c_{:(t+1)}\right.\right) = p\left(p\left(\kappa,\mu|c_{:t}\right)\left|c_{:(t+1)}\right.\right) = p\left(p\left(\kappa,\mu|c_{:t}\right)|s_t,c_t\right) = p\left(b_t^{\kappa,\mu}|\,s_t,c_t\right) \quad (49)$$

$$\propto p\left(b_t^{\kappa,\mu},c_t|\,s_t\right) = p\left(c_t\,|s_t,b_t^{\kappa,\mu}\right)p\left(b_t^{\kappa,\mu}|\,s_t\right) = p\left(c_t\,|s_t,b_t^{\kappa,\mu}\right)b_t^{\kappa,\mu} \quad (50)$$

$$= \mathbb{E}_{(\kappa_i,\mu_i)\sim b_t^{\kappa,\mu}}\left[\mu_i\left(a_t|s_t\right)\cdot R^{\kappa_i}(r_t|s_t,a_t)\cdot P^{\kappa_i}(s_{t+1}|s_t,a_t)\right]\cdot b_t^{\kappa,\mu} \quad (51)$$

is the posterior over the policy-constrained MDPs given the context $c_{:(t+1)}$, $\mathcal{A}$, $\mathcal{R}$ and $\widehat{H}$ are the same action space, reward space, and planning horizon as the finite-horizon BAMDP M^+, respectively, $\overline{P}^+ : \overline{\mathcal{S}}^+ \times \mathcal{A} \times \mathcal{R} \to \Delta\left(\overline{\mathcal{S}}^+\right)$ is the transition function: $\forall \bar{s}_t^+, \bar{s}_{t+1}^+ \in \overline{\mathcal{S}}^+, a_t \in \mathcal{A}, r_t \in \mathcal{R}$, where denoting $\bar{s}_t^+ = (s_t, b_t^{\kappa,\mu})$ and $\bar{s}_{t+1}^+ = \left(s_{t+1}, \tilde{b}_{t+1}^{\kappa,\mu}\right)$,

$$\overline{P}^+\left(\bar{s}_{t+1}^+\left|\bar{s}_t^+,a_t,r_t\right.\right) = \overline{P}^+\left(s_{t+1},\tilde{b}_{t+1}^{\kappa,\mu}\middle|\,s_t,b_t^{\kappa,\mu},a_t,r_t\right) \quad (52)$$

$$= \overline{P}^+\left(s_{t+1}\,|s_t,b_t^{\kappa,\mu},a_t,r_t\right)P^+\left(\tilde{b}_{t+1}^{\kappa,\mu}\middle|\,s_t,b_t^{\kappa,\mu},c_t\right) \quad (53)$$

$$= \mathbb{E}_{(\kappa_i,\mu_i)\sim p\left(b_t^{\kappa,\mu}|s_t,a_t,r_t\right)}\left[P^{\kappa_i}(s_{t+1}|s_t,a_t)\right]\cdot\mathbb{1}\left[\tilde{b}_{t+1}^{\kappa,\mu} = p(b_t^{\kappa,\mu}|s_t,c_t)\right], \quad (54)$$

$\overline{P}_0^+ : \Delta\left(\overline{\mathcal{S}}^+\right)$ is the initial hyper-state distribution, i.e., a deterministic initial hyper-state is

$$\bar{s}_0^+ = (s_0, b_0^{\kappa,\mu}) = (s_0, p(\kappa,\mu)) \in \overline{\mathcal{S}}^+, \quad (55)$$

and $\overline{R}^+ : \overline{\mathcal{S}}^+ \times \mathcal{A} \to \Delta\left(\mathcal{R}\right)$ is the reward distribution: $\forall \bar{s}^+ = (s, b^{\kappa,\mu}) \in \overline{\mathcal{S}}^+, a \in \mathcal{A}, r \in \mathcal{R}$,

$$\overline{R}^+\left(r|\bar{s}^+,a\right) = \overline{R}^+\left(r\,|s,b^{\kappa,\mu},a\right) = \mathbb{E}_{(\kappa_i,\mu_i)\sim b^{\kappa,\mu}}\left[R^{\kappa_i}(r|s,a)\right]. \quad (56)$$

In a transformed BAMDP $\overline{M}^+$, the overall belief is about the task-dependent behavior policies, transition function, and reward function, which are constant for a given task. A meta-policy on $\overline{M}^+$ is $\bar{\pi}^+ : \overline{\mathcal{S}}^+ \to \Delta\left(\mathcal{A}\right)$ prescribes a distribution over actions for each hyper-state. The agent's objective is now to find a meta-policy on hyper-states $\bar{\pi}^+$ that maximizes the expected return in the policies-constrained BAMDP,

$$\mathcal{J}_{\overline{M}^+}\left(\bar{\pi}^+\right) = \mathbb{E}_{a_t\sim\bar{\pi}^+\left(\cdot\left|\bar{s}_t^+\right.\right),r_t\sim\overline{R}^+\left(\cdot\left|\bar{s}_t^+,a_t\right.\right),s_{t+1}^+\sim\overline{P}^+\left(\cdot\left|\bar{s}_t^+,a_t\right.\right)}\left[\sum_{t=0}^{\widehat{H}-1} r_t\right] \quad (57)$$

$$= \mathbb{E}_{\kappa_i\sim p(\kappa)}\left[\sum_{j=0}^{N-1}\mathbb{E}_{a_t\sim\bar{\pi}^+\left(\cdot\left|\bar{s}_{jH+t}^+\right.\right),r_t\sim R^{\kappa_i}(\cdot|s_t,a_t),s_{t+1}\sim P^{\kappa_i}(\cdot|s_t,a_t)}\left[\sum_{t=0}^{H-1} r_t\right]\right]. \quad (58)$$

For any meta-policy on hyper-states $\bar{\pi}^+$, denote the corresponding meta-policy on history contexts $\hat{f}_{\bar{\pi}^+} : \mathcal{C} \to \Delta\left(\mathcal{A}\right)$, i.e., $\forall t \in \left[\widehat{H}-1\right], \forall c_{:t} \in \mathcal{C}_t$, s.t., $\hat{f}_{\bar{\pi}^+}\left(\cdot|c_{:t}\right) = \bar{\pi}^+\left(\cdot\left|\bar{s}_t^+\right.\right)$, where $\bar{s}_t^+ = (s_t, b_t^{\kappa,\mu}) = (s_t, p\left(\kappa,\mu|c_{:t}\right))$, and we have

$$\mathcal{J}_{\widehat{M}}\left(\hat{f}_{\bar{\pi}^+}\right) = \mathbb{E}_{\kappa_i\sim p(\kappa)}\left[\sum_{j=0}^{N-1}\mathbb{E}_{a_t\sim\hat{f}_{\bar{\pi}^+}\left(\cdot\left|c_{:(jH+t)}\right.\right),r_t\sim R^{\kappa_i}(\cdot|s_t,a_t),s_{t+1}\sim P^{\kappa_i}(\cdot|s_t,a_t)}\left[\sum_{t=0}^{H-1} r_t\right]\right] \quad (59)$$

$$= \mathcal{J}_{\overline{M}^+}\left(\bar{\pi}^+\right). \quad (60)$$

Fact 1. *Transformed BAMDPs implicitly require the agent to filter out out-of-distribution episodes.*

Proof. During online adaptation, RL agents construct a hyper-state $\bar{s}_t^+ = \left(s_t, \bar{b}_t\right)$ from the context history and perform a meta-policy $\bar{\pi}^+\left(a_t\left|\bar{s}_t^+\right.\right)$. The new belief $\bar{b}_t$ accounts for the uncertainty of task MDPs and behavior policies. However, the context history may be conflict with the uncertainty

estimation of behavior policies, i.e., RL agents cannot update their beliefs $\bar{b}_t$ when they have observed an event that they believe to have probability zero.

In the example shown in Figure 4, RL agents can take an action a_1 and receive a reward of 0 during online adaptation. After obsevering an action a_1, the posterior belief will be $p\left(\kappa_1, \mu_1 \left|\bar{b}_t\right.\right) = 1$, in which it is contradictory because the reward function of a_1 is 1 in the task κ_1. □

Lemma 1. *In a MDP M, for each behavior policy μ and batch-constrained policy π, collect a dataset D and the gap between approximate offline policy evaluation $\mathcal{J}_{\mathcal{D}}(\pi)$ and accurate policy evaluation $\mathcal{J}_M(\pi)$ will asymptotically approach to 0, as the offline dataset D grows.*

Proof. From a given dataset D, an abstract MDP $\hat{M}$ can be estimated (Fujimoto et al., 2019; Yin & Wang, 2021; Szepesvári, 2022). According to concentration bounds, the estimated transition and reward function will asymptotically approach M (Yin & Wang, 2021). Then, using the simulation lemma (Alekh Agarwal, 2017; Szepesvári, 2022), the gap between $\mathcal{J}_{\mathcal{D}}(\pi)$ and $\mathcal{J}_M(\pi)$ will asymptotically approach to 0, as the offline dataset D grows. □

Theorem 2. *In a transformed BAMDP $\overline{M}^+$, for each task-dependent behavior policies $[\mu]$ and batch-constrained meta-policy $\bar{\pi}^+$, the data distribution induced by $\bar{\pi}^+$ and $[\mu]$ matches after filtering out out-of-distribution episodes in online adaptation. Besides, the policy evaluation of $\bar{\pi}^+$ in offline meta-training and online adaptation will be asymptotically consistent, as the offline dataset grows.*

Proof. We assume feasible Bayesian belief updating in this proof. At first, $\forall s_t^+, a_t$, s.t. $p_{\overline{M}^+}^{\bar{\pi}^+}\left(\bar{s}_t^+, a_t\right) > 0$, we aim to prove

$$\mathbb{P}_{\overline{M}^+}\left(r_t, s_{t+1} \left|\bar{s}_t^+, a_t\right.\right) = \mathbb{P}_{\overline{M}^+,[\mu]}\left(r_t, s_{t+1} \left|\bar{s}_t^+, a_t\right.\right). \tag{61}$$

Since $\bar{\pi}^+$ is batch-constrained policy by $[\mu]$, if $p_{\overline{M}^+}^{\bar{\pi}^+}\left(\bar{s}_t^+, a_t\right) > 0$, we have $p_{\overline{M}^+}^{[\mu]}\left(\bar{s}_t^+, a_t\right) > 0$. Then,

$$\begin{aligned}
\mathbb{P}_{\overline{M}^+}\left(r_t, s_{t+1} \left|\bar{s}_t^+, a_t\right.\right) &= \overline{R}^+\left(r_t \left|\bar{s}_t^+, a_t\right.\right) \cdot \overline{P}^+\left(s_{t+1} \left|\bar{s}_t^+, a_t\right.\right) && (62)\\
&= \mathbb{E}_{\kappa_i \sim b_t^{\kappa,\mu}}\left[\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \left|s_t, a_t\right.\right)\right] && (63)\\
&= \mathbb{E}_{(\kappa_i,\mu_i) \sim b_t^{\kappa,\mu}}\left[\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \left|s_t, a_t\right.\right)\right] && (64)
\end{aligned}$$

where $b_t^{\kappa,\mu}$ is defined in Eq. (48) and

$$p\left(\kappa_i, \mu_i \left|b_t^{\kappa,\mu}\right.\right) = \mathbb{E}_{\kappa_i \sim p(\kappa), \mu_i \sim p(\mu|\kappa_i)}\left[\mathbb{E}_{c_{:t} \sim p_{\overline{M}^+}(c_{:t}|\kappa_i,\mu_i)}\left[\mathbb{1}\left[b_t^{\kappa,\mu} = p(\kappa,\mu|c_{:t})\right]\right]\right], \tag{65}$$

where $p_{\overline{M}^+}\left(c_{:t}|\kappa_i, \mu_i\right)$ is defined in Eq. (9). According to Eq. (39)

$$\begin{aligned}
&\mathbb{P}_{\overline{M}^+,[\mu]}\left(r_t, s_{t+1} \left|\bar{s}_t^+, a_t\right.\right) && (66)\\
&\propto \mathbb{E}_{\kappa_i \sim p(\kappa), \mu_i \sim p(\mu|\kappa_i)}\left[\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \left|s_t, a_t\right.\right) \cdot p_{\overline{M}^+}\left(\bar{s}_t^+ \left|\kappa_i, \mu_i\right.\right)\right] && (67)\\
&= \mathbb{E}_{\kappa_i \sim p(\kappa), \mu_i \sim p(\mu|\kappa_i)}\left[\mathbb{E}_{c_{:t} \sim p_{\overline{M}^+}(c_{:t}|\kappa_i,\mu_i)}\left[\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \left|s_t, a_t\right.\right) \cdot \mathbb{1}\left[b_t^{\kappa,\mu} = p(\kappa,\mu|c_{:t})\right]\right]\right] && (68)\\
&= \mathbb{E}_{(\kappa_i,\mu_i) \sim b_t^{\kappa,\mu}}\left[\mathbb{P}_{\kappa_i}\left(r_t, s_{t+1} \left|s_t, a_t\right.\right)\right] && (69)\\
&= \mathbb{P}_{\overline{M}^+}\left(r_t, s_{t+1} \left|\bar{s}_t^+, a_t\right.\right). && (70)
\end{aligned}$$

Thus, the data distribution induced by $\bar{\pi}^+$ and $[\mu]$ matches. Directly use Lemma 1 in a transformed BAMDP $\overline{M}^+$, in which $\overline{M}^+$ is a belief MDP, a type of MDP. Therefore, the policy evaluation of $\bar{\pi}^+$ in offline meta-training and online adaptation will be asymptotically consistent, as the offline dataset grows. □

B Hyper-Parameter Settings

Environment Settings. Table 6 shows hyper-parameter settings for the task sets used in our experiments. Most hyper-parameters are adopted from previous works (Li et al., 2020; Mitchell et al.,

Table 6: Environment parameter settings.

Environment	Episode Length	# of Adaptation Episodes	# of Tasks	# of Trajectories per Task
All Meta-World Envs	500	10	50	45
Cheetah-Vel	200	10	100	45
Point-Robot	20	20	100	45
Point-Robot-Sparse	20	20	100	45

Table 7: Detailed hyper-parameter settings for GCC.

Hyper-Parameter	Hyper-Parameter Values
batch size	256
meta batch size	16
learning rates for dual critic	1e-4
learning rates for all other components	3e-4
network structure for all components	three fully connected layers with 200 units
optimizer	adam
discount	0.99
latent size	20
reward scale	100 for point envs, 1 for all other envs
n_{it}	1/2 of total adaptation episodes
n_0^z	1/2 of total adaptation episodes
$n_{\tilde{z}}$	5
n_e	1

2021). For all task sets, 80% of the tasks are meta-training tasks, and the remaining 20% tasks are meta-testing tasks.

GCC hyper-parameter settings. Table 7 shows GCC's hyper-parameter settings. Most hyper-parameters are adopted from FOCAL (Li et al., 2020). We set n_e to 1 as the evaluation environments are all nearly deterministic.

C Additional Experiment Results

Table 8 and Table 9 shows baselines' online adaptation and offline performance on all 50 Meta-World ML1 task sets, respectively. GCC significantly outperforms baselines with online adaptation, and achieves better or comparable performance to baselines with offline adaptation.

Table 8: Comparison between baselines with online adaptation and GCC. For Meta-World tasks, "-V2" is omitted for brevity.

Environment	GCC	FOCAL	MACAW
Coffee-Push	**1.26** ± 0.13	0.66 ± 0.07	0.01 ± 0.01
Faucet-Close	**1.12** ± 0.01	1.06 ± 0.02	0.07 ± 0.01
Door-Unlock	**1.11** ± 0.02	0.97 ± 0.03	0.11 ± 0.01
Plate-Slide-Side	**1.07** ± 0.08	0.70 ± 0.14	0.00 ± 0.00
Faucet-Open	**1.05** ± 0.02	1.01 ± 0.02	0.08 ± 0.04
Button-Press-Wall	**1.04** ± 0.04	0.99 ± 0.06	0.02 ± 0.00
Plate-Slide	**1.01** ± 0.03	0.83 ± 0.09	0.01 ± 0.00
Door-Close	**0.99** ± 0.00	0.97 ± 0.01	0.00 ± 0.00
Drawer-Close	**0.99** ± 0.02	**0.96** ± 0.04	0.53 ± 0.50
Plate-Slide-Back-Side	**0.97** ± 0.02	0.77 ± 0.20	0.02 ± 0.01
Door-Lock	**0.97** ± 0.01	0.90 ± 0.02	0.25 ± 0.11
Window-Open	**0.96** ± 0.02	0.81 ± 0.07	0.15 ± 0.11
Door-Open	**0.96** ± 0.02	0.78 ± 0.13	0.06 ± 0.01
Plate-Slide-Back	**0.96** ± 0.02	0.58 ± 0.06	0.21 ± 0.17
Window-Close	**0.94** ± 0.01	0.79 ± 0.01	0.54 ± 0.44
Reach-Wall	**0.93** ± 0.05	0.53 ± 0.18	0.82 ± 0.02
Dial-Turn	**0.91** ± 0.05	0.84 ± 0.09	0.00 ± 0.00
Handle-Press-Side	**0.91** ± 0.02	0.79 ± 0.10	0.51 ± 0.41
Handle-Pull	**0.90** ± 0.02	0.67 ± 0.03	0.00 ± 0.00
Handle-Press	**0.88** ± 0.05	**0.87** ± 0.02	0.28 ± 0.10
Reach	**0.85** ± 0.03	0.62 ± 0.05	0.63 ± 0.04
Lever-Pull	**0.85** ± 0.02	0.73 ± 0.07	0.20 ± 0.16
Hammer	**0.84** ± 0.06	0.59 ± 0.07	0.10 ± 0.01
Drawer-Open	**0.82** ± 0.06	0.64 ± 0.10	0.11 ± 0.02
Sweep	**0.77** ± 0.04	0.32 ± 0.08	0.20 ± 0.20
Button-Press	**0.74** ± 0.08	**0.68** ± 0.14	0.02 ± 0.01
Stick-Push	**0.73** ± 0.09	0.46 ± 0.15	0.17 ± 0.17
Coffee-Button	**0.73** ± 0.14	**0.66** ± 0.16	0.15 ± 0.13
Push-Wall	**0.71** ± 0.15	0.43 ± 0.06	0.23 ± 0.18
Shelf-Place	**0.70** ± 0.18	0.32 ± 0.11	0.01 ± 0.01
Basketball	**0.64** ± 0.15	0.41 ± 0.24	0.00 ± 0.00
Hand-Insert	**0.63** ± 0.04	0.29 ± 0.07	0.02 ± 0.01
Sweep-Into	**0.61** ± 0.06	0.33 ± 0.05	0.00 ± 0.00
Button-Press-Topdown	**0.57** ± 0.11	0.45 ± 0.06	0.38 ± 0.36
Peg-Unplug-Side	**0.56** ± 0.07	0.19 ± 0.09	0.00 ± 0.00
Assembly	**0.55** ± 0.13	0.28 ± 0.05	0.33 ± 0.01
Push	**0.55** ± 0.10	0.34 ± 0.14	0.28 ± 0.19
Bin-Picking	0.53 ± 0.16	0.31 ± 0.21	**0.66** ± 0.11
Push-Back	**0.52** ± 0.05	0.16 ± 0.04	0.00 ± 0.00
Box-Close	**0.51** ± 0.11	0.15 ± 0.09	0.36 ± 0.11
Soccer	**0.44** ± 0.04	0.11 ± 0.03	**0.38** ± 0.31
Button-Press-Topdown-Wall	**0.43** ± 0.03	**0.40** ± 0.07	0.05 ± 0.02
Disassemble	**0.42** ± 0.14	0.26 ± 0.04	0.05 ± 0.00
Coffee-Pull	**0.40** ± 0.05	0.23 ± 0.04	0.19 ± 0.12
Stick-Pull	**0.32** ± 0.06	0.17 ± 0.07	0.00 ± 0.00
Peg-Insert-Side	**0.30** ± 0.04	0.08 ± 0.03	0.00 ± 0.00
Pick-Place-Wall	0.28 ± 0.12	0.09 ± 0.04	**0.39** ± 0.25
Pick-Out-Of-Hole	0.26 ± 0.25	0.16 ± 0.16	**0.59** ± 0.06
Pick-Place	**0.20** ± 0.03	0.07 ± 0.02	0.05 ± 0.05
Handle-Pull-Side	**0.14** ± 0.04	**0.13** ± 0.09	0.00 ± 0.00
Average	**0.73** ± 0.07	0.53 ± 0.08	0.18 ± 0.09
Cheetah-Vel	**-171.52** ± 21.96	-287.70 ± 30.62	-233.97 ± 23.46
Point-Robot	**-5.10** ± 0.26	-15.38 ± 0.95	-14.61 ± 0.98
Point-Robot-Sparse	**7.78** ± 0.64	0.83 ± 0.37	0.00 ± 0.00

Table 9: Comparison between baselines with offline adaptation and GCC. For Meta-World tasks, "-V2" is omitted for brevity.

Environment	GCC	FOCAL with Expert Context	MACAW with Expert Context
Coffee-Push	**1.26** ± 0.13	0.50 ± 0.06	**1.14** ± 0.27
Faucet-Close	**1.12** ± 0.01	1.07 ± 0.02	1.01 ± 0.01
Door-Unlock	**1.11** ± 0.02	0.96 ± 0.03	0.99 ± 0.04
Plate-Slide-Side	**1.07** ± 0.08	0.75 ± 0.09	0.91 ± 0.09
Faucet-Open	**1.05** ± 0.02	**1.00** ± 0.02	0.99 ± 0.01
Button-Press-Wall	**1.04** ± 0.04	0.98 ± 0.05	**0.99** ± 0.01
Plate-Slide	**1.01** ± 0.03	**1.00** ± 0.03	0.67 ± 0.07
Door-Close	**0.99** ± 0.00	0.97 ± 0.01	0.92 ± 0.05
Drawer-Close	**0.99** ± 0.02	0.96 ± 0.04	**1.00** ± 0.01
Plate-Slide-Back-Side	**0.97** ± 0.02	0.90 ± 0.07	0.80 ± 0.05
Door-Lock	**0.97** ± 0.01	0.88 ± 0.04	**0.97** ± 0.03
Window-Open	**0.96** ± 0.02	**0.93** ± 0.05	**0.98** ± 0.02
Door-Open	**0.96** ± 0.02	0.90 ± 0.02	**0.99** ± 0.02
Plate-Slide-Back	**0.96** ± 0.02	**0.93** ± 0.01	0.55 ± 0.11
Window-Close	0.94 ± 0.01	0.73 ± 0.02	**1.00** ± 0.01
Reach-Wall	**0.93** ± 0.05	**0.91** ± 0.05	0.82 ± 0.02
Dial-Turn	0.91 ± 0.05	0.84 ± 0.08	**0.98** ± 0.01
Handle-Press-Side	**0.91** ± 0.02	0.87 ± 0.04	0.82 ± 0.10
Handle-Pull	**0.90** ± 0.02	0.70 ± 0.05	**0.95** ± 0.05
Handle-Press	**0.88** ± 0.05	**0.79** ± 0.08	0.56 ± 0.19
Reach	**0.85** ± 0.03	**0.83** ± 0.05	0.64 ± 0.08
Lever-Pull	0.85 ± 0.02	0.76 ± 0.03	**0.97** ± 0.07
Hammer	**0.84** ± 0.06	0.78 ± 0.04	0.40 ± 0.18
Drawer-Open	0.82 ± 0.06	0.73 ± 0.11	**0.98** ± 0.01
Sweep	0.77 ± 0.04	0.74 ± 0.02	**0.98** ± 0.01
Button-Press	**0.74** ± 0.08	0.63 ± 0.09	**0.71** ± 0.04
Stick-Push	**0.73** ± 0.09	0.14 ± 0.09	**0.67** ± 0.09
Coffee-Button	**0.73** ± 0.14	**0.61** ± 0.20	0.21 ± 0.11
Push-Wall	0.71 ± 0.15	**0.90** ± 0.12	**0.96** ± 0.09
Shelf-Place	**0.70** ± 0.18	0.57 ± 0.13	0.55 ± 0.03
Basketball	**0.64** ± 0.15	0.49 ± 0.17	0.47 ± 0.18
Hand-Insert	**0.63** ± 0.04	**0.64** ± 0.09	0.20 ± 0.09
Sweep-Into	**0.61** ± 0.06	**0.64** ± 0.09	0.00 ± 0.00
Button-Press-Topdown	0.57 ± 0.11	0.48 ± 0.11	**0.92** ± 0.04
Peg-Unplug-Side	**0.56** ± 0.07	**0.57** ± 0.10	0.18 ± 0.10
Assembly	0.55 ± 0.13	**0.64** ± 0.03	0.36 ± 0.02
Push	0.55 ± 0.10	**0.98** ± 0.13	0.86 ± 0.02
Bin-Picking	0.53 ± 0.16	**0.61** ± 0.12	**0.63** ± 0.11
Push-Back	**0.52** ± 0.05	**0.52** ± 0.16	0.15 ± 0.09
Box-Close	**0.51** ± 0.11	**0.56** ± 0.08	0.35 ± 0.11
Soccer	0.44 ± 0.04	0.45 ± 0.03	**0.59** ± 0.11
Button-Press-Topdown-Wall	**0.43** ± 0.03	**0.40** ± 0.06	**0.43** ± 0.06
Disassemble	0.42 ± 0.14	0.23 ± 0.05	**0.50** ± 0.06
Coffee-Pull	0.40 ± 0.05	**0.58** ± 0.11	0.45 ± 0.11
Stick-Pull	**0.32** ± 0.06	0.18 ± 0.06	**0.27** ± 0.09
Peg-Insert-Side	0.30 ± 0.04	**0.52** ± 0.08	0.25 ± 0.04
Pick-Place-Wall	**0.28** ± 0.12	0.13 ± 0.07	**0.21** ± 0.16
Pick-Out-Of-Hole	0.26 ± 0.25	0.27 ± 0.27	**0.59** ± 0.08
Pick-Place	0.20 ± 0.03	0.29 ± 0.11	**0.72** ± 0.09
Handle-Pull-Side	0.14 ± 0.04	0.09 ± 0.05	**0.94** ± 0.08
Average	**0.73** ± 0.07	0.67 ± 0.07	0.68 ± 0.07
Cheetah-Vel	**-171.52** ± 21.96	-156.07 ± 23.22	-292.92 ± 36.66
Point-Robot	**-5.10** ± 0.26	-4.68 ± 0.18	-19.60 ± 1.15
Point-Robot-Sparse	**7.78** ± 0.64	8.37 ± 0.67	0.00 ± 0.00