Meta-Reinforcement Learning via Exploratory Task Clustering

Zhendong Chu[1], **Renqin Cai**[2], **Hongning Wang**[1]

[1]Department of Computer Science, University of Virginia
[2]Meta
zc9uy@virginia.edu, renqincai@meta.com, hw5x@virginia.edu

Abstract

Meta-reinforcement learning (meta-RL) aims to quickly solve new RL tasks by leveraging knowledge from prior tasks. Previous studies often assume a single-mode homogeneous task distribution, ignoring possible structured heterogeneity among tasks. Such an oversight can hamper effective exploration and adaptation, especially with limited samples. In this work, we harness the structured heterogeneity among tasks via clustering to improve meta-RL, which facilitates knowledge sharing at the cluster level. To facilitate exploration, we also develop a dedicated cluster-level exploratory policy to discover task clusters via divide-and-conquer. The knowledge from the discovered clusters helps to narrow the search space of task-specific policy learning, leading to more sample-efficient policy adaptation. We evaluate the proposed method on environments with parametric clusters (e.g., rewards and state dynamics in the MuJoCo suite) and non-parametric clusters (e.g., control skills in the Meta-World suite). The results demonstrate strong advantages of our solution against a set of representative meta-RL methods.

Introduction

Conventional reinforcement learning (RL) is notorious for its high sample complexity, which often requires tremendous amount of interactions with an environment to learn a performing policy for a new task (Huai et al. 2020; Yao, Cai, and Wang 2021). Inspired by the learning process of humans, meta-reinforcement learning (meta-RL) is proposed to quickly learn new tasks by leveraging knowledge shared by related tasks (Finn, Abbeel, and Levine 2017; Duan et al. 2016; Wang et al. 2016). The key research question in meta-RL is task modeling for identifying transferable knowledge among tasks. For example, Finn, Abbeel, and Levine (2017) proposed to learn a set of shared meta parameters which are used to initialize the local policy when a new task arrives. Duan et al. (2016) and Wang et al. (2016) trained an RNN encoder to characterize prior tasks according to the interaction history in those tasks.

Little attention has been paid to the structures in the transferable knowledge resulted from task distributions. Aforementioned methods implicitly assume tasks follow a uni-modal distribution, and thus the knowledge, once identified, can be broadly shared across all tasks. However, heterogeneity among tasks is not rare in practice. It therefore dwarfs simple sharing of global knowledge, but instead imposes subtle structures for identifying relatedness among tasks at a finer granularity, e.g., groups of tasks. For instance, the general skills required for the Go game and Gomoku game are related, such as familiarity with the board layout and stone colors. But to achieve mastery in either game, policies must acquire and internalize game-specific knowledge/rules to effectively navigate subsequent matches. For example, experience about competing against different human players in Go games can be shared within, but not over to Gomoku games. This heterogeneity motivates us to formulate a more delicate but also more general meta-RL setting where tasks are originated from various but a finite number of distributions, i.e., *tasks are clustered*. Hence, knowledge that benefits learning in new tasks becomes cluster-specific. We refer to this as *structured heterogeneity* among tasks, and propose to explicitly model it to facilitate cluster-level knowledge sharing[1].

Structured heterogeneity among tasks has been studied in supervised meta-learning (Yao et al. 2019; Zhang et al. 2023); but it is a lot more challenging to be handled in meta-RL, where the key bottleneck is how to efficiently discover task relatedness in a population of RL tasks. Different from supervised learning tasks where static task-specific data is available for task relatedness inference before any learning starts, observations about RL tasks are collected by an agent's interactions with the task environment. As a result, successfully adapting an RL policy to a new task depends on accurate profiling of the task, which however is elicited by the policy itself. Task inference becomes a major bottleneck of sample efficiency in meta-RL (Liu et al. 2021). Previous methods (Duan et al. 2016; Chu et al. 2023) focus on task embedding learning under the uni-modal task distribution assumption, which are inefficient to infer clustered tasks. But structured heterogeneity provides new opportunities for efficient task inference: instead of directly identifying the new task, the coarse-grained cluster membership can be first inferred with a few observations; within the located task cluster, task inference can be performed in a designated search space, i.e., *divide-and-conquer* task inference.

[1]We do not assume the knowledge in different clusters is exclusive, and thus each cluster can still contain overlapping knowledge, e.g., motor skills in locomotion tasks.

To realize our idea of utilizing structured heterogeneity among tasks in meta-RL, we develop MILET: **M**eta re**I**nforcement **L**earning via **E**xploratory Task clus**T**ering. To the best of our knowledge, we are the first to propose a method for improving sample efficiency in meta-RL by utilizing cluster structures in the task distribution. Specifically, we perform cluster-based variational inference (CBVI) (Rao et al. 2019; Dilokthanakul et al. 2016) to infer the cluster of a new task according to its ongoing trajectory. To facilitate cluster inference, at the meta-train phase, we optimize a dedicated exploration policy based on a divide-and-conquer strategy: it first quickly explores the task's cluster assignment, and then refines its task modeling in the narrowed search space given the identified cluster. An exploitation policy is then trained to maximize the task rewards based on the refined task model from the exploration trajectory. We compare MILET against a rich set of state-of-the-art meta-RL solutions on various MuJoCo environments (Todorov, Erez, and Tassa 2012) with varying cluster structures in both reward and state dynamics. To test the generality of MILET, we further evaluate it on environments characterized by non-parametric cluster structures among tasks, i.e., the Meta-World tasks (Yu et al. 2020). The experiment results confirm MILET can effectively discover clusters among tasks and then benefit fast adaptation to new tasks.

The main contributions of this paper are three-fold,

1. We present MILET, a novel approach to improve meta-RL by explicitly modeling cluster structures inherent in the task distribution.
2. We introduce a dedicated cluster-level exploratory policy, which employs a divide-and-conquer strategy, ensuring robust and effective discovery of task clusters.
3. We evaluate MILET on a rich set of environments with both parametric and non-parametric task clusters. The empirical results prove the effectiveness of MILET.

Related Work

Task modeling in meta-learning. Task modeling is important to realize fast adaptation in new tasks in meta learning. Finn, Abbeel, and Levine (2017) first proposed the model-agnostic meta learning (MAML) aiming to learn a shared model initialization, i.e., the meta model, given a population of tasks. MAML does not explicitly model tasks, but it expects the meta model to be only a few gradient updates away from all tasks. Later, an array of methods extend MAML by explicitly modeling tasks using given training data under the supervised meta-learning setting (Lee and Choi 2018; Vuorio et al. 2019). Yao et al. (2019) adopted a hierarchical task clustering structure, which enables cluster-specific meta model. Such a design encourages the solution to capture locally transferable knowledge inside each cluster, similar to our MILET model. However, task information is not explicitly available in meta-RL: since the true reward/state transition functions are not accessible to the agent, the agent needs to interact with the environment to collect observations about the tasks, while maximizing its return from the interactions. MILET models posterior distribution of a task's cluster assignment based on its ongoing trajectory; better yet, it is designed to behave exploratorily to quickly identify tasks' clustering structures, and then refine the task modeling in the narrowed search space conditional on the identified cluster.

Exploration in meta-reinforcement learning. Exploration plays an important role in meta-RL, as the agent can only learn from its interactions with the environment. In gradient-based meta-RL (Finn, Abbeel, and Levine 2017), the local policy is trained on the trajectories collected by the meta policy, and thus the exploration for task structure is not explicitly handled. Stadie et al. (2018) and Rothfuss et al. (2018) computed gradients with respect to the sampling distribution of the meta policy, in addition to the collected trajectories. Gupta et al. (2018) also extended MAML by using learnable latent variables to control different exploration behaviors. The context-based meta-RL algorithms (Duan et al. 2016; Wang et al. 2016) automatically learn to trade off exploration and exploitation by learning a policy conditioned on the current context. Zintgraf et al. (2020) explicitly provided the task uncertainty to the policy to facilitate exploration. Zhang et al. (2021) and Liu et al. (2021) developed a separate exploration policy by maximizing the mutual information between task ids and inferred task embeddings. However, because all the aforementioned methods operate under the uni-modal assumption about the task distribution, their exploration strategy also becomes inferior to profile a given task under a heterogeneous task distribution. MILET first explores to identify the cluster of a task, which is expected to require fewer samples than detailed task identification; then the agent can explore task information within a refined search space for better sample efficiency.

Background

Meta-reinforcement learning. We consider a family of Markov decision processes (MDPs)[2] $p(\mathcal{M})$, where an MDP $M_i \sim p(\mathcal{M})$ is defined by a tuple $M_i = (\mathcal{S}, \mathcal{A}, R_i, T_i, T_{i,0}, \gamma, H)$ with $\mathcal{S}$ denoting its state space, $\mathcal{A}$ as its action space, $R_i(r_{t+1}|s_t, a_t)$ as its reward function, $T_i(s_{t+1}|s_t, a_t)$ as its state transition function, $T_{i,0}(s_0)$ as its initial state distribution, γ as a discount factor, and H as the length of an episode. The index i represents the task id, which is provided to agents in some works (Zhang et al. 2021; Liu et al. 2021; Rakelly et al. 2019). We consider a more general setting where the task id is not provided to the agent (Zintgraf et al. 2020), as in general we should not expect the task id to encode any task-related information. Tasks sampled from $p(\mathcal{M})$ typically differ in the reward and/or transition functions. In each task, we run a *trial* consisting of $1 + N$ episodes (Duan et al. 2016). Following the evaluation settings in previous works (Finn, Abbeel, and Levine 2017; Liu et al. 2021; Rothfuss et al. 2018), the first episode in a trial is reserved as an *exploration* episode to gather information for task modeling, and an agent is evaluated by the returns in the following N *exploitation* episodes.

Inside a trial, we denote the agent's interaction with the MDP at time step n as $\tau_n = \{s_n, a_n, r_n, s_{n+1}\}$, and $\tau_{:t} =$

[2]The terms of environment, task and MDP are used interchangeably in this paper, when no ambiguity is incurred.

$\{s_0, a_0, r_0, ..., s_t\}$ denotes the interaction history collected before time t. In the exploration episode, an agent should form the most informative trajectory τ^ψ by rolling out an exploration policy π_ψ parameterized by ψ. In the exploitation episodes, the agent executes the exploitation policy π_ϕ parameterized by ϕ (in some prior work, π_ψ and π_ϕ are the same (Zintgraf et al. 2020)) conditioned on τ^ψ and, optionally, the history collected in the exploitation episodes τ^ϕ. The returns in exploitation episodes are computed as,

$$\mathcal{J}(\pi_\psi, \pi_\phi) = \mathbb{E}_{M_i \sim p(\mathcal{M}), \tau^\psi \sim \pi_\psi} \left[\sum_{t=0}^{N \times H} R_i\big(\pi_\phi(\tau^\psi; \tau^\phi_{:t})\big) \right], \tag{1}$$

where $R_i\big(\pi_\phi(\tau^\psi; \tau^\phi_{:t})\big)$ is the return of π_ϕ conditioned on τ^ψ and $\tau^\phi_{:t}$ at time step t in task M_i.

Clustered RL tasks. In this paper, we consider a more general and realistic setting, where the task distribution is multi-modal and thus forms a mixture,

$$p(\mathcal{M}) = \sum_{c=1}^{C} w_c \cdot p_c(\mathcal{M}), \tag{2}$$

where C is the number of mixing components (i.e., clusters) and w_c is the corresponding weight of component c, such that $\sum_{c=1}^{C} w_c = 1$. Thus, every task is sampled as follows,

1. Sample a cluster c according to the multinomial distribution of $Mul(w_1, ..., w_C)$;
2. Sample a reward function R or a transition function T or both from $p_c(\mathcal{M})$.

The knowledge shared in different clusters could be different. For example, two clusters of distinct target positions can exist in a navigational environment, e.g., top-left vs., bottom-right. The knowledge about how an agent reaches the top-left target positions in the first cluster cannot help tasks in the second cluster; but it is crucial for learning different tasks in the first cluster. In this example, when handling a new task, a good exploration strategy should first recognize the task cluster (i.e., to move top-left or bottom-right), as it is much easier to recognize than individual tasks, and then identify the specific target position in the corresponding region of the map. This coarse-to-fine identification allows more efficient exploration of task information.

Methodology

In this section, we present MILET in detail, which consists of two complementary components. First, we introduce how to infer population-level task structures using the collected trajectories via cluster-based variational inference (CBVI). Then, we explain the exploration policy trained by the exploration-driven reward, which is designed to quickly identify the cluster assignment of a new task. At a high level, in each task MILET first executes the exploration policy to collect the coarse-grained cluster information; then it adapts the task policy with the help of inferred posterior cluster distribution. The architecture of MILET is shown in Figure 1.

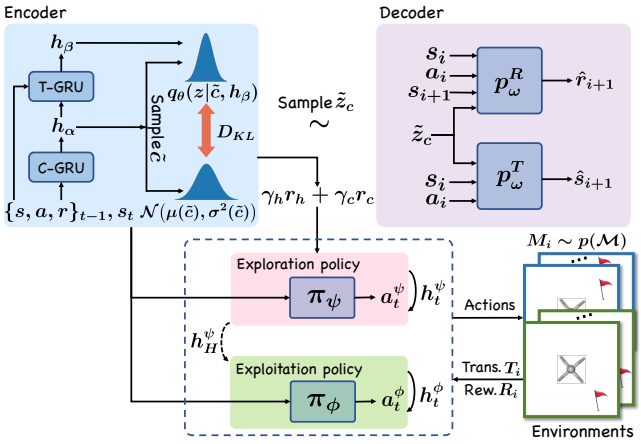


Figure 1: MILET architecture. The encoder processes ongoing trajectories and performs CBVI for $q_\theta(z|c, h_\beta)$. The exploration policy π_ψ is trained to find the most certain cluster assignment c when interacting with the environment. The explored information is passed to the exploitation policy π_ϕ to facilitate fast adaptation in task M_i.

Cluster-based Variational Inference with Consistency Regularization

Since the reward and transition functions are unknown to the agent, we estimate a latent random variable c_i to infer the cluster assignment of current task $M_i \sim p_c(\mathcal{M})$. Based on c_i, we infer another latent random variable z_i carrying task-level information, i.e., z_i suggests the reward/transition functions that define the task. For simplicity, we first drop the subscript i in this section, as we will only use one task as an example to illustrate our model design.

In meta-RL, all information about a given task can be encoded by z. But inferring z can be sample inefficient, as the task space can be very large. Thanks to the structured heterogeneity among tasks, inferring a task's cluster assignment c can be more sample efficient, since we should expect a much smaller number of task clusters than the number of tasks. Once c is identified, z can be more efficiently identified, i.e., divide and conquer. Hence, in MILET, when a new task arrives, we decode its characteristics by the posterior distribution $p(z, c|\tau_{:t}) = p(z|\tau_{:t}, c)p(c|\tau_{:t})$ with respect to the interaction history up to time t. The inferred task information z_c, which refers to z conditioned on c, is then provided to the policy $\pi_{\psi/\phi}(a_t|s_t, z_c)$.

Exact posterior of $p(z, c|\tau_{:t})$ defined by Eq.(2) is intractable. Instead, we learn an approximated variational posterior $q_\theta(z, c|\tau_{:t}) = q_\theta(z|\tau_{:t}, c)q_\theta(c|\tau_{:t})$, in which we estimate two dependent inference networks and collectively denote their parameters as θ. On top of the inference networks, we learn a decoder p_ω to reconstruct the collected trajectories. The whole framework is trained by maximizing the following objective,

$$\mathbb{E}_{\rho_\pi(\mathcal{M}, \tau^+)} \Big[\log p(\tau^+|\pi) \Big], \tag{3}$$

where ρ_π is the distribution of trajectories induced by the policies $\pi = \{\pi_\psi, \pi_\phi\}$ within the task, and $\tau^+ = \{\tau^\psi, \tau^\phi\}$

denotes all trajectories collected in a trial, the length of which is denoted as $H^+ = (N+1)H$. We choose to use trajectories from both exploration and exploitation episodes to best leverage information about the same underlying MDP. We omit the dependencies on π to simplify our notations in later discussions. Instead of optimizing the intractable objective in Eq.(3), we optimize its evidence lower bound (ELBO) w.r.t. the approximated posterior $q_\theta(z, c|\tau_{:t})$ estimated via Monte Carlo sampling (Rao et al. 2019) (full derivation can be found in **Appendix A**),

$$\begin{aligned} ELBO_t = \mathbb{E}_\rho \Bigg[& \overbrace{\mathbb{E}_{q_\theta(z,c|\tau_{:t})}\big[\ln p_\omega(\tau^+|\tilde{z}_c)\big]}^{\text{cluster-specific reconstruction likelihood}} \\ & - \overbrace{\mathbb{E}_{q_\theta(c|\tau_{:t})}\big[\mathrm{KL}(q_\theta(z|c,\tau_{:t}) \parallel p_\omega(z|c))\big]}^{\text{cluster-specific regularization}} \\ & - \overbrace{\mathrm{KL}(q_\theta(c|\tau_{:t}) \parallel p(c))}^{\text{cluster regularization}} \Bigg], \end{aligned} \tag{4}$$

where $p_\omega(z|c) = \mathcal{N}\big(\mu_\omega(c), \sigma^2_\omega(c)\big)$ is a learnable cluster-specific prior, which is different from the simple Gaussian prior used in single-mode VAE (Kingma and Welling 2013). This prior allows MILET to capture unique characteristics of each cluster. $p_\omega(z|c)$'s parameters are included in ω since the cluster structure is also part of the environment. $\tilde{z}_c$ is the latent variable sampled from $q_\theta(z|c,\tau_{:t}) = \mathcal{N}\big(\mu_\theta(c,\tau_{:t}), \sigma^2_\theta(c,\tau_{:t})\big)$, using the reparameterization trick (Kingma and Welling 2013). $q_\theta(c|\tau_{:t})$ outputs the approximated posterior cluster distribution given $\tau_{:t}$[3]. $p(c)$ is the prior cluster distribution of tasks; when no specific prior knowledge is known about the task, we choose a fixed non-informative multinomial distribution for it. Intuitively, if discrete structures (i.e., clusters) exist in the task distribution, a uniform $q_\theta(c|\tau_{:t})$ will cause low reconstruction likelihood; thus collapsed posterior, i.e., clustering, is preferred.

Similar to (Zintgraf et al. 2020), the first term $\ln p_\omega(\tau^+|\tilde{z}_c)$ in Eq.(4) can be further factorized as,

$$\begin{aligned} \ln p_\omega(\tau^+|\tilde{z}_c) = & \ln p(s_0|\tilde{z}_c) + \sum_{i=0}^{H^+-1} \Big[\ln p_\omega(s_{i+1}|s_i, a_i, \tilde{z}_c) \\ & + \ln p_\omega(r_{i+1}|s_i, a_i, s_{i+1}, \tilde{z}_c) \Big], \end{aligned}$$

where $p(s_0|\tilde{z}_c)$ is the initial state distribution in a task, and we consider it as a constant by assuming identical distribution of the initial states across clusters. The second and third terms are likelihood derived from the decoders for transition and reward functions. The density functions of $p_\omega(s_{i+1}|s_i, a_i, \tilde{z}_c)$ and $p_\omega(r_{i+1}|s_i, a_i, s_{i+1}, \tilde{z}_c)$ are difficult to estimate in continuous state and action spaces. Following (Zhang et al. 2021; Babaeizadeh et al. 2018), we use L2 distance to approximate the log-likelihood functions.

In the inference networks $q_\theta(z|\tau_{:t}, c)$ and $q_\theta(c|\tau_{:t})$, we follow (Duan et al. 2016; Zintgraf et al. 2020) to encode the history $\tau_{:t}$ by Gated Recurrent Units (GRUs) (Chung et al. 2014; Cai et al. 2021; Wu, Cai, and Wang 2020). We propose a stacked GRU structure (shown in Figure 1) to differentiate the information for cluster and task inference in the hidden space. Specifically, we set a task-GRU (T-GRU) and a cluster-GRU (C-GRU), both of which encode the history $\tau_{:t}$, but with different levels of granularity. T-GRU is set to capture fine-grained task-specific patterns in the history, as it is optimized to reconstruct trajectories of a specific task. C-GRU captures coarser-grained patterns beyond tasks, as it is set to help T-GRU reconstruct all trajectories within a cluster. To realize this difference, the output h_β of T-GRU is only provided to $q_\theta\big(z|h_\beta(\tau_{:t}, h_\alpha), c\big)$, while the output h_α of C-GRU is passed to both cluster inference $q_\theta\big(c|h_\alpha(\tau_{:t})\big)$ and task inference $q_\theta\big(z|h_\beta(\tau_{:t}, h_\alpha), c\big)$. This also reflects our dependency assumption about the task structure: cluster assignment determines tasks. We denote $h = \{h_\alpha, h_\beta\}$, which is passed across episodes in a trial.

The trajectory data is incrementally collected by the agent in meta-RL, which brings both challenges and opportunities for cluster inference. First, inside a trial, the inference improves as more observations are collected, which means the agent's belief about the ongoing task could change thereby. This is problematic, since the cluster inference result should stay consistent within a given task, no matter how trajectory changes over episodes. We attribute this property as *in-trial* consistency, which is measured by $\mathrm{KL}(q(c|\tau_{:t_1}) \parallel q(c|\tau_{:t_2}))$, where t_1 and t_2 refer to two arbitrary timestamps in a trial. We enforce the notion of cluster inference consistency via the following regularizer,

$$\mathcal{L}_\mathrm{I} = \frac{1}{H^+ - 1} \sum_{t=0}^{H^+-1} \mathrm{KL}\big(q_\theta(c|\tau_{:t}) \parallel q_\theta(c|\tau_{:t+1})\big). \tag{5}$$

Similarly, since the cluster-specific prior $p_\omega(z|c)$ is learnable, the task inference can become inconsistent if $p_\omega(z|c)$ changes drastically across training epochs. More seriously, oscillation in the inference of latent variable z can cause the collapse of policy training, as tasks across clusters might be assigned with the same latent variable z across different training epochs. We conclude it as the *prior* consistency requirement and enforce it via the following regularization,

$$\mathcal{L}_\mathrm{P} = \frac{1}{C} \sum_{c=1}^{C} \mathrm{KL}\big(p_\omega(z|c) \parallel p_\mathrm{tgt}(z|c)\big), \tag{6}$$

where $p_\mathrm{tgt}(z|c)$ is a target network and its parameters are the same as $p_\omega(z|c)$ but updated in a much slower pace. We finally obtain the objective in CBVI as follows,

$$\mathcal{J}(\theta, \omega) = \mathbb{E}_{p(\mathcal{M})}\Big[\sum_{t=0}^{H+} ELBO_t - \lambda_\mathrm{I}\mathcal{L}_\mathrm{I} - \lambda_\mathrm{P}\mathcal{L}_\mathrm{P} \Big], \tag{7}$$

where λ_I and λ_P are hyper-parameters to control the strength of two regularizers.

Exploration via Reducing Inference Uncertainty

In MILET, policy adaptation in a new task has two objectives: (1) explore cluster structure; (2) explore task-specific information to solve the task. As we explained before,

[3] We use the Gumbel-softmax trick to simplify the calculation.

MILET follows a divide-and-conquer principle to realize these two objectives, which is implemented by learning two separate policies as shown in Figure 1. One takes exploratory behaviors to collect cluster and task information, i.e., the exploration policy π_ψ. The other is optimized to solve the task with the collected information, i.e., the exploitation policy π_ϕ.

We train a dedicated exploration policy to provide a good basis for task-solving, where cluster structures provide informative hints about task relatedness. The quality of exploration is evaluated by two principles. First, whether the trajectory of an exploration episode can reduce the uncertainty of cluster inference. Second, whether the inference result is consistent. We conclude them as *certain* and *consistent* exploration. To realize these two principles, we introduce two intrinsic rewards to encourage certain and consistent inference results. First, we use the entropy of cluster inference network $q_\theta(c|\tau^{\psi}_{:H})$ to measure the *uncertainty* of the inferred cluster. For a new task, we look for trajectories that provide the most certain cluster inference. We formalize the objective as follows, omitting the subscript θ and ψ for simplicity,

$$H(q(c|\tau_{:H})) = -\mathbb{E}\big[\ln q(c|\tau_0) + \sum_{t=0}^{H-1} \ln \frac{q(c|\tau_{:t+1})}{q(c|\tau_{:t})}\big].$$

We then define an intrinsic reward of each action by telescoping the second term similar to (Zhang et al. 2021; Liu et al. 2021),

$$\begin{aligned} r_h(a_t) =& \mathbb{E}\big[\ln \frac{q(c|\tau_{:t+1} = [s_{t+1}; a_t; r_t; \tau_{:t}])}{q(c|\tau_{:t})}\big] \\ =& H(q(c|\tau_{:t})) - H(q(c|\tau_{:t+1})). \end{aligned}$$

This reward favorites actions which can reduce the entropy of cluster inference; and therefore, a trajectory leading to a consistent cluster inference is preferred. To more explicitly measure the divergence between the posterior cluster distributions in two steps, we define another reward encouraging consistent cluster inference,

$$r_c(a_t) = -\mathrm{KL}(q(c|\tau_{:t}) \parallel q(c|\tau_{:t+1})).$$

Intuitively, given the inferred cluster, the exploration policy can focus on identifying task-level information within a narrowed search space, i.e., divide-and-conquer. We define the following composed reward to encourage this coarse-to-fine exploration behavior,

$$r_e(a_t) = r(a_t) + \gamma_h(t) r_h(a_t) + \gamma_c(t) r_c(a_t), \tag{8}$$

where $r(a_t)$ is the environment reward. $\gamma_h(t)$ and $\gamma_c(t)$ are two temporal decaying functions,

$$\gamma_h(t) = b_h - a_h \exp(-s_h(H-t)), \tag{9}$$

$$\gamma_c(t) = -b_c + a_c \exp(-s_c(H-t)), \tag{10}$$

where $\{a, b, s\}_{h,c}$ are hyper-parameters controlling the rate of decay. $\gamma_h(t)$ should gradually decrease to 0, which encourages the policy to find a certain cluster at the early stage. $\gamma_c(t)$ gradually increases from a negative value to positive. At the early stage, a negative $\gamma_c(t)$ encourages the policy to try different clusters. Later, a positive $\gamma_c(t)$ enforces the policy to stick to the current cluster and focuses more on discovering task information by maximizing raw rewards. We provide a detailed discussion on the design of Eq.(10) in **Appendix D** and present the comparison of different exploration strategies in **Appendix E**.

Finally, the exploitation policy π_ϕ inherits the hidden state $h_H^{\pi_\psi}$, which encodes knowledge collected by the exploration policy, and is then trained to maximize the expected reward defined in Eq.(1). The detailed pseudo-codes of meta-train and meta-test phases for MILET are shown in **Appendix B**.

Experiments

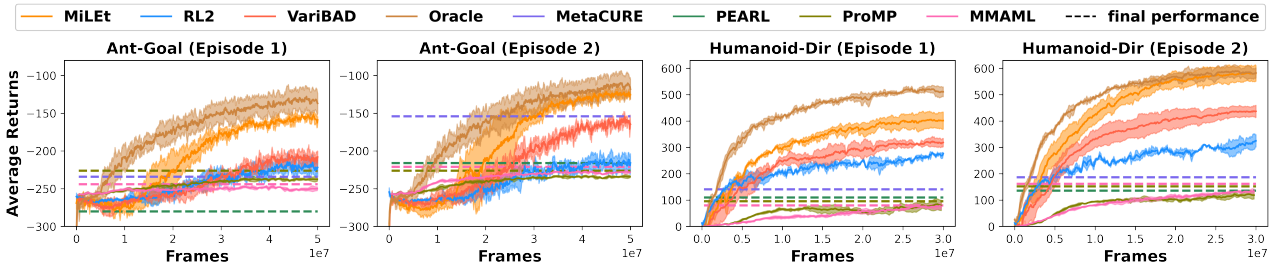
In this section, we conduct extensive experiments to study the following research questions: (1) Can MILET achieve better performance than state-of-the-art meta-RL algorithms by exploring structured heterogeneity in the task distribution? (2) Can MILET effectively discover cluster structures in both rewards and state dynamics? (3) How does the number of clusters affect the final performance of MILET? (4) Is MILET capable of clustering tasks with non-parametric cluster structures, e.g., the Meta-World tasks? Due to space limit, we defer the comprehensive ablation study of MILET to **Appendix E**. We further evaluate MILET on handling out-of-distribution clusters of tasks (**Appendix I**) and more difficult environments with partial rewards (**Appendix J**).

Results on MuJoCo Environments

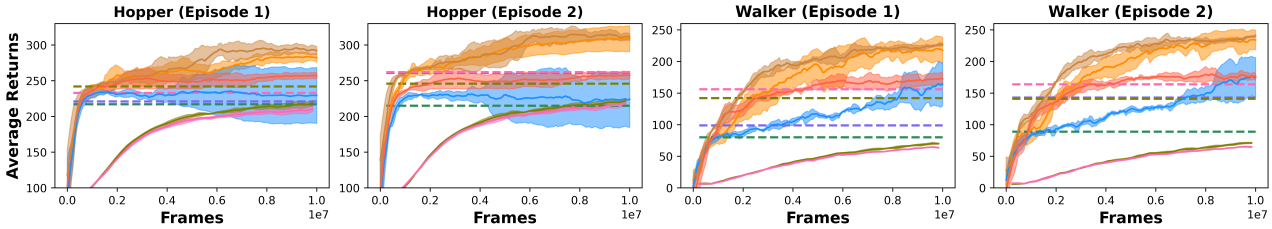
Environment setup. We evaluated MILET on two continuous control tasks with *clustered reward functions*, simulated by MuJoCo (Todorov, Erez, and Tassa 2012). In **Ant-Goal**, the ant robot is set to move to a predetermined goal position. We created 4 clusters of the goal positions in 4 different centered areas. In **Humanoid-Dir**, the human-like robot is controlled to move towards different target directions. We created 4 clusters by distributing target directions along 4 farthest apart directions in a 2D space. We also created environments with *clustered transition functions* by adopting two movement environments **Hopper-Rand-Params** and **Walker-Rand-Params**, also simulated by MuJoCo. The physical parameters of the robot, including *body mass*, *damping on degrees of freedom*, *body inertia* and *geometry friction*, were manipulated to realize different transition functions of the robot's movement. The hopper and walker robots are set to move smoothly under different parameter settings. We created 4 clusters by manipulating one of the parameters at a time and keeping the others to the default parameters. More details can be found in **Appendix C**.

Baseline setup. We compared MILET with several representative meta-RL baselines, including **RL**^{2} (Duan et al. 2016), **PEARL** (Rakelly et al. 2019), **VariBAD** (Zintgraf et al. 2020), **MetaCURE** (Zhang et al. 2021), **ProMP** (Rothfuss et al. 2018) and **MMAML** (Vuorio et al. 2019). We also included an **Oracle** model, where we trained a separate VariBAD model for each ground-truth cluster. We used implementations of baselines provided by the original papers.

For each environment, we created 500 tasks for meta-train and hold out 32 new tasks for meta-test. We report the performance on test tasks during the meta-train phase. In the meta-test phase, we executed 2 episodes in each new

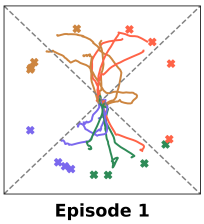


(a) Environments with clustered reward functions.

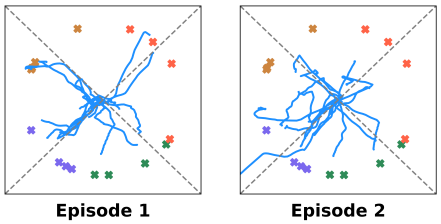


(b) Environments with clustered state transition functions.

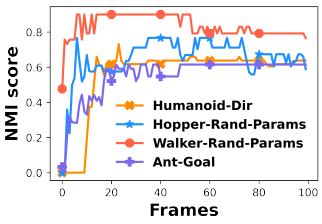
Figure 2: Average test performance for 2 episodes on MuJoCo environments.



(a) MiLEt traces.



(b) VariBAD traces.



(c) NMI score in exploration.

Figure 3: Qualitative analysis of MILET. (a) Traces of MILET on the meta-test tasks of Ant-Goal. Cross marks represent goal positions, and the colors represent the clusters assigned by MILET. The dashed lines suggest the optimal traces to the centers of ground-truth clusters. (b) Traces of VariBAD on the same meta-test tasks of Ant-Goal. The traces are in the same color as VariBAD is unaware of clusters. (c) NMI of MILET's inferred clusters in the exploration episode of meta-test tasks.

task. For algorithms with an explicit exploration policy, i.e., MILET and MetaCURE, we run their exploration policy in the first episode and exploitation policy in the second episode. We trained MILET via Proximal Policy Optimization (PPO) (Schulman et al. 2017) and set the default cluster number C to 4. Because PEARL and MetaCURE are based on off-policy algorithms (Haarnoja et al. 2018), they need less frames of data to converge in meta-train. We terminated them once the algorithm was converged and reported the final performance obtained by the moment. We report the averaged performance over 3 random seeds. More implementation details can be found in **Appendix D**.

Results and analysis. Figure 2 shows the test performance of all evaluated meta-RL algorithms. We also provide qualitative analysis in Figure 3, including visualization of the models' behaviors and the clustering performance of MILET in the exploration episode, measured by the normalized mutual information score (NMI).

First, we clearly observed Oracle performed the best in both episodes. By directly utilizing shared knowledge within correct clusters, Oracle is able to fast adapt to individual tasks. It shows the necessity of accurate cluster modeling for fast adaptation. MILET showed significant improvement against baselines in the second episode in testing, approaching the performance of Oracle. Interestingly, we can observe even though the first episode of MILET was reserved for exploration, it still performed comparably to other methods in all four different environment setups. In the first episode, MILET behaved exploratorily to find the most probable cluster of the current task, and thus its traces in Figure 3a look like spirals from the starting point. VariBAD is also designed to explore by uncertainty in task inference, but its traces were close to random walk at the early stage, which is less effective. In Figure 3c, we can observe the NMI scores of the MILET's inferred tasks have almost converged in 20 steps, which means the cluster inference became stable in an early stage and can thereby provide the agent helpful cluster-level information to gain fine-grained task information. This also explains how MILET obtained comparable performance in the first episode. In the second episode,

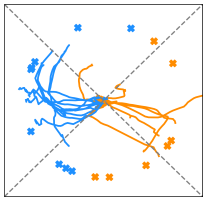
	Ant-Goal	Ant-U
VariBAD	-168.6±9.6	-162.4±9.2
MILET-2	-132.3±7.6	-128.6±8.8
MILET-4	-125.4±5.1	-113.7±4.8
MILET-6	-123.6±4.4	-99.7±5.2
MILET-8	-124.2±4.7	-117.9±5.7
MILET-10	-128.6±5.2	-142.7±10.4

Table 1: Results on Ant-Goal and Ant-U.

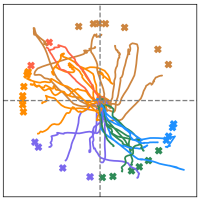
with cultivated task information, MILET is able to move towards the targets directly, showing significant improvements against baselines. MetaCURE guides the exploration by task IDs, which in fact provides more information of environment than what MILET can access. However, the exploration empowered by task IDs does not explicitly explore the coarser but useful information at the cluster level. Both ProMP and MMAML are gradient-based methods, we found they need 5 times samples to converge, thus we also reported the final performance. Importantly, MMAML, tailored for multi-modal tasks, faces challenges during exploration. It relies heavily on its meta-policy to explore task-specific information and then formulates task embeddings. If exploration is suboptimal, the final performance suffers.

Influence of the Number of Clusters

We also studied how the number of clusters C set by the agent influences the final performance, especially when there is a mismatch between the ground-truth cluster size and C set by the agent. We set C to different values and denote it in suffixes of MILET. We additionally created a set of tasks on Ant-Goal, where the goal positions were uniformly sampled. We denote it as **Ant-U**.



(a) MILET-2 on Ant-G.



(b) MILET-6 on Ant-U.

Figure 4: Traces of MILET-2 and -6 in exploration episodes. Colors represent the assigned clusters.

The average final returns are shown in Table 1. Interestingly, we observe MILET can perform well even though there is no explicit cluster structure in Ant-U. By looking into the detailed trajectories, we found MILET segmented the circle into different parts as shown in Figure 4b such that knowledge from nearby tasks can be effectively shared. VariBAD mistakenly assumed all tasks can share knowledge and thus failed seriously. When C is set smaller than the ground-truth number of clusters, MILET-2 discovered more general structures (as shown in Figure 4a). However, transferable knowledge within such structures is limited as distinct clusters are merged, causing the performance drop. Also, it does not mean more clusters than necessary is helpful, as less knowledge could be shared in each cluster. In Ant-U, MILET-8 and -10 generated unnecessary clusters, and cluster assignments are mixed at the boundary of adjacent clusters (visualized in **Appendix G**). Such inaccurate cluster modeling causes ineffective exploration and knowledge sharing, leading to degenerated performance.

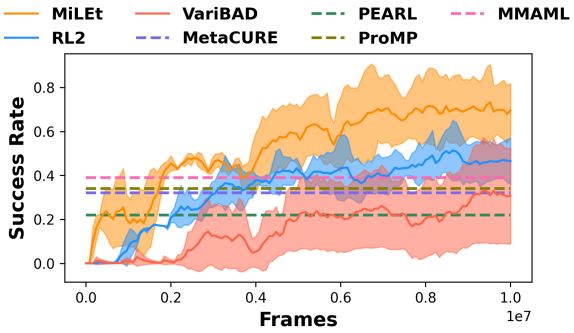


Figure 5: Meta-World results.

Results on Meta-World Environments

We also evaluated MILET on a challenging task suite Meta-World (Yu et al. 2020), which includes a variety of robot arm control tasks. Each task needs specific skills to solve, e.g., pushing and pulling, and different tasks share different degrees of relatedness. It is crucial to differentiate tasks and use corresponding skills. There are also variants inside each task, e.g., positions of objects and targets. We considered a combination of 8 set of tasks: {Push, Reach, Drawer-Close, Button-Press, Plate-Slide, Plate-Slide-Side, Plate-Slide-Back-Side, Plate-Slide-Back}. We held out 8 variants of each task for testing and present results in Figure 5. MILET achieved higher success rates by correctly recognizing tasks and using task-specific skills.

Conclusion & Future Work

In this paper, we present MILET, a cluster-based solution to improve meta-RL by utilizing the structured heterogeneity of tasks. MILET is able to discover clustered task structures in a population of RL-tasks and enable cluster-level knowledge sharing to new tasks. To quickly identify the cluster assignment of new tasks, MILET learns a dedicated exploratory policy based on a divide-and-conquer strategy.

Several extensions of MILET are worth exploring in our future work. First, more complicated priors can be considered to enable new features. For example, the Dirichlet process prior can be used to automatically identify number of clusters in the task distribution, and possibly detect out-of-distribution tasks in the meta-test phase. Second, MILET can be combined with skill-based RL methods (Pertsch, Lee, and Lim 2021; Nam et al. 2022) to learn cluster-level skills, which will form a new basis for meta-RL, e.g., each task is modeled as a skill mixture, with different task clusters reflecting varied skill distributions.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work was supported by NSF IIS-2007492 and NSF IIS-1838615.

References

Babaeizadeh, M.; Finn, C.; Erhan, D.; Campbell, R. H.; and Levine, S. 2018. Stochastic Variational Video Prediction. In *International Conference on Learning Representations*.

Cai, R.; Wu, J.; San, A.; Wang, C.; and Wang, H. 2021. Category-aware collaborative sequential recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 388–397.

Chu, Z.; Wang, H.; Xiao, Y.; Long, B.; and Wu, L. 2023. Meta Policy Learning for Cold-Start Conversational Recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 222–230.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Dilokthanakul, N.; Mediano, P. A.; Garnelo, M.; Lee, M. C.; Salimbeni, H.; Arulkumaran, K.; and Shanahan, M. 2016. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.

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

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Gupta, A.; Mendonca, R.; Liu, Y.; Abbeel, P.; and Levine, S. 2018. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Huai, M.; Sun, J.; Cai, R.; Yao, L.; and Zhang, A. 2020. Malicious attacks against deep reinforcement learning interpretations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 472–482.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lee, Y.; and Choi, S. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, 2927–2936. PMLR.

Liu, E. Z.; Raghunathan, A.; Liang, P.; and Finn, C. 2021. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International conference on machine learning*, 6925–6935. PMLR.

Nam, T.; Sun, S.-H.; Pertsch, K.; Hwang, S. J.; and Lim, J. J. 2022. Skill-based Meta-Reinforcement Learning. In *International Conference on Learning Representations*.

Pertsch, K.; Lee, Y.; and Lim, J. 2021. Accelerating Reinforcement Learning with Learned Skill Priors. In *Conference on Robot Learning*, 188–204. PMLR.

Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; and Quillen, D. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, 5331–5340. PMLR.

Rao, D.; Visin, F.; Rusu, A.; Pascanu, R.; Teh, Y. W.; and Hadsell, R. 2019. Continual unsupervised representation learning. *Advances in Neural Information Processing Systems*, 32.

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

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

Stadie, B. C.; Yang, G.; Houthooft, R.; Chen, X.; Duan, Y.; Wu, Y.; Abbeel, P.; and Sutskever, I. 2018. Some Considerations on Learning to Explore via Meta-Reinforcement Learning. *ArXiv*, abs/1803.01118.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

Vuorio, R.; Sun, S.-H.; Hu, H.; and Lim, J. J. 2019. Multimodal model-agnostic meta-learning via task-aware modulation. *Advances in neural information processing systems*, 32.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Wu, J.; Cai, R.; and Wang, H. 2020. Déjà vu: A contextualized temporal attention mechanism for sequential recommendation. In *Proceedings of The Web Conference 2020*, 2199–2209.

Yao, F.; Cai, R.; and Wang, H. 2021. Reversible action design for combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:2102.07210*.

Yao, H.; Wei, Y.; Huang, J.; and Li, Z. 2019. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, 7045–7054. PMLR.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, 1094–1100. PMLR.

Zhang, J.; Li, A.; Tang, M.; Sun, J.; Chen, X.; Zhang, F.; Chen, C.; Chen, Y.; and Li, H. 2023. Fed-cbs: A heterogeneity-aware client sampling mechanism for federated learning via class-imbalance reduction. In *International Conference on Machine Learning*, 41354–41381. PMLR.

Zhang, J.; Wang, J.; Hu, H.; Chen, T.; Chen, Y.; Fan, C.; and Zhang, C. 2021. Metacure: Meta reinforcement learning with empowerment-driven exploration. In *International Conference on Machine Learning*, 12600–12610. PMLR.

Zintgraf, L.; Shiarlis, K.; Igl, M.; Schulze, S.; Gal, Y.; Hofmann, K.; and Whiteson, S. 2020. VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning. In *International Conference on Learning Representations*.