DynaMITE-RL: A Dynamic Model for Improved Temporal Meta-Reinforcement Learning

Anthony Liang [1 2] **Guy Tennenholtz** [2] **Chih-wei Hsu** [2] **Yinlam Chow** [2] **Erdem Bıyık** [1] **Craig Boutilier** [2]

Abstract

We introduce *DynaMITE-RL*, a meta-reinforcement learning (meta-RL) approach to approximate inference in environments where the latent state evolves at varying rates. We model episode sessions—parts of the episode where the latent state is fixed—and propose three key modifications to existing meta-RL methods: consistency of latent information within sessions, session masking, and prior latent conditioning. We demonstrate the importance of these modifications in various domains, ranging from discrete Gridworld environments to continuous-control and simulated robot assistive tasks, demonstrating that DynaMITE-RL significantly outperforms state-of-the-art baselines in sample efficiency and inference returns.

1. Introduction

Markov decision processes (MDPs) (Bertsekas, 2012) provide a general framework in reinforcement learning (RL), and can be used to model sequential decision problems in a variety of domains, e.g., recommender systems (RSs), robot and autonomous vehicle control, and healthcare (Jannach et al., 2021; Ie et al., 2019; Cao et al., 2020; Yu et al., 2021; Liu et al., 2020; Biyik et al., 2019). MDPs assume a static environment with fixed transition probabilities and rewards (Bellman, 1957). In many real-world systems, however, the dynamics of the environment are intrinsically tied to latent factors subject to temporal variation. While nonstationary MDPs are special instances of partially observable MDPs (POMDPs) (Kaelbling et al., 1998), in many applications these latent variables change infrequently, i.e. the latent variable remains fixed for some duration before changing. One class of problems exhibiting this DLCMDP structure is recommender systems, where a user's preferences are a latent variable which gradually evolves over time (Jawaheer et al., 2014; Kim et al., 2023). For instance, a user may initially have a strong affinity for a particular genre (e.g., action movies), but their viewing habits could change over time, influenced by external factors such as trending movies, mood, etc. A robust system should adapt to these evolving tastes to provide suitable recommendations. Another example is in manufacturing settings where industrial robots may experience unobserved gradual deterioration of their mechanical components affecting the overall dynamics of the system. Accurately modelling the changing dynamics caused by hardware degradation can help manufacturers optimize performance, cost, safety and equipment lifespan.

Our goal in this work is to leverage such a temporal structure to obviate the need to solve a fully general POMDP. To this end, we propose **Dyna**mic **M**odel for **I**mproved **T**emporal **Me**ta **R**einforcement **L**earning (DynaMITE-RL), a method designed to exploit the temporal structure of sessions, i.e., sequences of a trajectory in which the latent state is fixed. We formulate our problem as a *dynamic latent contextual MDP (DLCMDP)*, and identify three crucial elements needed to enable tractable and efficient policy learning in environments with the latent dynamics captured by a DLCMDP. First, we consider consistency of latent information, by exploiting time steps for which we have high confidence that the latent variable is constant. To do so, we introduce a consistency loss to regularize the posterior update model, providing better posterior estimates of the latent variable. Second, we enforce the posterior update model to learn the dynamics of the latent variable. This allows the trained policy to better infer, and adapt to, temporal shifts in latent context in unknown environments. Finally, we show that the variational objective in contemporary meta-RL algorithms, which attempts to reconstruct the entire trajectory, can hurt performance when the latent context is nonstationary. We modify this objective to reconstruct only the transitions within the same session (i.e., that share the same latent context).

Closest to our work is VariBAD (Zintgraf et al., 2020), a meta-RL (Beck et al., 2023) approach for learning a Bayes-optimal policy, enabling an agent to quickly adapt to a new environment with unknown dynamics and reward. VariBAD uses variational inference to learn a posterior update model

[1]Viterbi School of Engineering, University of Southern California Los Angeles, CA [2]Google Research. Correspondence to: Anthony Liang <anthony.liang@usc.edu>.

that approximates the belief over the transition and reward functions. It augments the state space with this belief to encode the agent's uncertainty during decision-making. Nevertheless, VariBAD and the Bayes-Adaptive MDP framework (Ross et al., 2007) assume the latent context is static *across an episode* and do not address settings with latent state dynamics. In this work, we focus on the dynamic latent state formulation of the meta-RL problem.

Our core contributions are as follows. (1) We introduce DynaMITE-RL, a meta-learning approach to handle environments with evolving latent context variables. (2) We introduce three key elements for learning an improved posterior update model: session consistency, modeling dynamics of latent context, and session reconstruction masking. (3) We validate our approach on a diverse set of challenging simulation environments and demonstrate significantly improved results over state-of-the-art baselines.

2. Background

We begin by reviewing relevant background including meta-RL and Bayesian RL. We also briefly summarize the VariBAD (Zintgraf et al., 2020) algorithm for learning Bayes-adaptive policies.

Meta-RL. The goal of meta-RL (Beck et al., 2023) is to quickly adapt an RL agent to an unseen test environment. Meta-RL assumes a distribution $p(\mathcal{T})$ over possible environments or *tasks*, and learns this distribution by repeatedly sampling batches of tasks during meta-training. Each task $\mathcal{T}_i \sim p(\mathcal{T})$ is described by an MDP $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{R}_i, \mathcal{P}_i, \gamma)$, where the state space $\mathcal{S}$, action space $\mathcal{A}$, and discount factor γ are shared across tasks, while $\mathcal{R}_i$ and $\mathcal{P}_i$ are task-specific reward and transition functions, respectively. The objective of meta-RL is to learn a policy that efficiently maximizes reward given a new task $\mathcal{T}_i \sim p(\mathcal{T})$ sampled from the task distribution at meta-test time. Meta-RL is a special case of a POMDP in which the unobserved variables are $\mathcal{R}$ and $\mathcal{P}$, which are assumed to be stationary throughout an episode.

Bayesian Reinforcement Learning (BRL). BRL (Ghavamzadeh et al., 2015) incorporates Bayesian inference to model agent uncertainty in decision making. In BRL, $\mathcal{R}$ and $\mathcal{P}$ are unknown a priori and treated as random variables with associated prior distributions. At time t, the *observed history* of states and actions is $\tau_{:t} = \{s_0, a_0, r_1, s_1, a_1, \ldots, r_t, s_t\}$, and the belief b_t represents the posterior over task parameters $\mathcal{R}$ and $\mathcal{P}$ given the transition history, i.e. $b_t \triangleq P(\mathcal{R}, \mathcal{P} \mid \tau_{:t})$. Given the initial belief $b_0(\mathcal{R}, \mathcal{P})$, the belief can be updated iteratively using Bayes' rule: $b_{t+1} = P(\mathcal{R}, \mathcal{P} \mid \tau_{:t+1}) \propto P(s_{t+1}, r_{t+1} \mid \tau_{:t}, \mathcal{R}, \mathcal{P}) b_t$. This Bayesian approach to RL can be formalized as a *Bayes-adaptive MDP (BAMDP)* (Duff, 2002). A BAMDP is an MDP over the *augmented state space* $S^+ = \mathcal{S} \times \mathcal{B}$, where $\mathcal{B}$ denotes the belief space. Given the augmented state $s_t^+ = (s_t, b_t)$, the transition function is given by $P^+(s_{t+1}^+ \mid s_t^+, a_t) = \mathbb{E}_{b_t}[\mathcal{P}(s_{t+1}|s_t, a_t)\delta(b_{t+1} = P(\mathcal{R}, \mathcal{P} \mid \tau_{:t+1})]$, and reward function is the expected reward given the belief, $R^+(s_t^+, a_t) = \mathbb{E}_{b_t}[\mathcal{R}(s_t, a_t)]$. The BAMDP formulation naturally resolves the exploration-exploitation tradeoff. A Bayes-optimal RL agent takes information-gathering actions to reduce its uncertainty in the MDP parameters while simultaneously maximizing its returns. However, for most interesting problems, solving the BAMDP—and even computing posterior updates—is intractable given the continuous and typically high-dimensional nature of its state space.

VariBAD. Zintgraf et al. (2020) approximate the Bayes-optimal solution by modeling uncertainty over the MDP parameters. These parameters are represented by a latent vector $m \in \mathbb{R}^d$, the posterior over which is $p(m \mid \tau_{:H})$, where H is the BAMDP horizon. Their VariBAD method uses a variational approximation, parameterized by ϕ, sharing the same structure, $q_\phi(m \mid \tau_{:t})$, and is conditioned on the observed history up to time t. Zintgraf et al. (2020) show that $q_\phi(m \mid \tau_{:t})$ approximates the belief b_t. In practice, $q_\phi(m \mid \tau_{:t})$ is represented by a Gaussian distribution $q_\phi(m \mid \tau_{:t}) = \mathcal{N}(\mu(\tau_{:t}), \Sigma(\tau_{:t}))$ where μ and Σ are recurrent neural networks (RNNs). The variational lower bound at time t is $\mathbb{E}_{q_\phi(m|\tau_{:t})}[\log p_\theta(\tau_{:H} \mid m)] - D_{KL}(q_\phi(m \mid \tau_{:t}) \parallel p_\theta(m))$. Intuitively, the first term reconstructs the trajectory and the second regularizes the variational posterior to a prior over the embeddings, typically a standard Gaussian. Importantly, the past trajectory $\tau_{:t}$ is used in the ELBO equation to infer the posterior belief at time t, which then decodes the entire trajectory $\tau_{:H}$, *including future transitions*. To approximately solve the BAMDP, the policy is a function of both the state and belief, $\pi(a_t \mid s_t, q_\phi(m \mid \tau_{:t}))$. The policy is trained using policy gradient, optimizing:

$$J(\pi) = \mathbb{E}_{\mathcal{R},\mathcal{P}} \mathbb{E}_\pi \left[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) \right] \tag{1}$$

where the first expectation is approximated by averaging over training environments and the RL agent is trained jointly with the variational autoencoder q_ϕ.

3. Dynamic Latent Contextual MDPs

A *dynamic latent contextual MDP (DLCMDP)* is given by $(\mathcal{S}, \mathcal{A}, \mathcal{M}, \mathcal{R}, T, \nu_0, H)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{M}$ is the *latent* context space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{M} \mapsto \Delta_{[0,1]}$ is a reward function, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{M} \mapsto \Delta_{\mathcal{S} \times \mathcal{M}}$ is a transition function, $\nu_0 \in \Delta_{\mathcal{S} \times \mathcal{M}}$ is an initial state distribution, $\gamma \in (0, 1)$ is a discount factor, and H is the (possibly infinite) horizon.

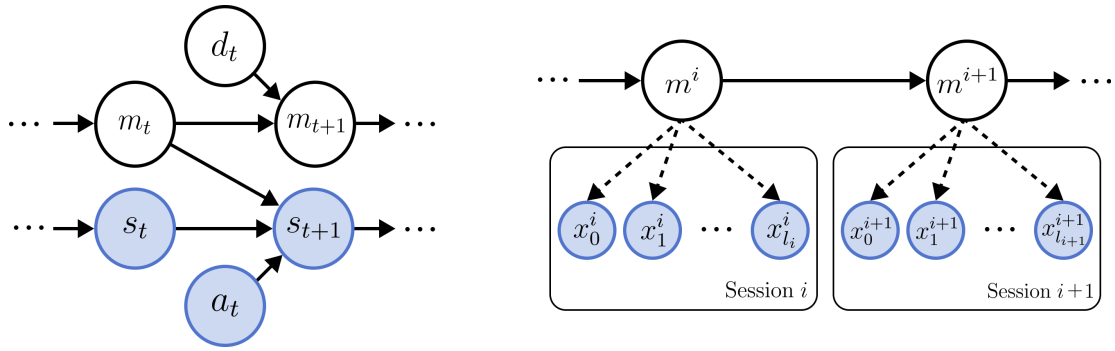

Figure 1. **(Left)** The graphical model for a DLCMDP. The transition dynamics of the environment follows $T(s_{t+1}, m_{t+1} \mid s_t, a_t, m_t)$. At every timestep t, an i.i.d. Bernoulli random variable, d_t, denotes the change in the latent context, m_t. Blue shaded variables are observed, whereas white shaded variables are latent. **(Right)** A realization of a DLCMDP episode. Each session i is governed by a latent variable m^i which is changing between sessions according to a fixed transition function, $T(m' \mid m)$. We denote l_i as the length of session i. The state-action pair (s_t^i, a_t^i) at timestep t in session i is summarized into a single observed variable, x_t^i. We emphasize that session terminations are not explicitly observed.

We assume an episodic setting in which each episode begins in a state-context pair $(s_0, m_0) \sim \nu_0$. At time t, the agent is at state s_t and context m_t, and has observed history $\tau_{:t} = \{s_0, a_0, r_1, \ldots, r_t, s_t\}$. Given the history, the agent selects an action $a_t \in \mathcal{A}$, after which the state and latent context transition according to $T(s_{t+1}, m_{t+1} \mid s_t, a_t, m_t)$, and the agent receives a reward sampled from $\mathcal{R}(s_t, a_t, m_t)$. Throughout this process, the context m_t is latent (i.e., *not observed* by the agent).

DLCMDPs embody the causal independence depicted by the graphical model in Figure 1. Letting $\Omega = \{d_t\}_{t=0}^{H-1}$ denote a sequence of i.i.d. Bernoulli random variables, we assume that

$$\begin{aligned} &T(s_{t+1} = s', m_{t+1} = m' \mid s_t = s, a_t = a, m_t = m) \\ &= T(s' \mid s, a, m)\mathbb{1}\{m' = m, d_t = 0\}P(d_t = 0) \\ &\quad + \nu_0(s' \mid m')T(m' \mid m)\mathbb{1}\{d_t = 1\}P(d_t = 1). \end{aligned}$$

Here, d_t defines a random variable at which a transition occurs in m_t. We refer to sub-trajectories between changes in the latent context as *sessions*, which may vary in length. At the start of a new session, a new state and a new latent context are sampled based on the distribution ν_0. Each session is itself an MDP governed by some unknown task parameters or latent context $m \in \mathcal{M}$ which changes stochastically between sessions according to dynamics $T(m' \mid m)$. For notational simplicity we sometimes use index i to denote the i^{th} session in a trajectory, and m^i the respective latent context of that session. We emphasize that sessions switching times are latent random variables.

Notice that DLCMDPs are more general than latent MDPs (Steimle et al., 2021; Kwon et al., 2021), in which the latent context is fixed throughout the entire episode; this corresponds to $d_t \equiv 0$. Moreover, DLCMDPs are closely related to POMDPs; letting $d_t \equiv 1$, a DLCMDP reduces to a POMDP with state space $\mathcal{M}$, observation space $\mathcal{S}$, and observation function ν_0. As a consequence DLCMDPs are as general as POMDPs, rendering them very expressive. That said, the specific temporal structure of DLCMDPs allows us to devise an efficient algorithm that exploits the transition dynamics of the latent context, improving learning efficiency. Finally, DLCMDPs are also related to DCMDPs (Tennenholtz et al., 2023), though DCMDPs assume contexts are observed, and focus on aggregated context dynamics.

We aim to learn a policy $\pi(a_t \mid s_t, m_t)$ which maximizes the expected return $J(\pi)$ in an unseen test environment per Eq. (1). As in BAMDPs, the optimal DLCMDP Q-function satisfies the Bellman equation; $\forall s^+ \in \mathcal{S}^+, a \in \mathcal{A}$:

$$\begin{aligned} Q(s^+, a) &= R^+(s^+, a) \\ &\quad + \gamma \sum_{s^{+'} \in \mathcal{S}^+} P^+(s^{+'} \mid s^+, a) \max_{a'} Q(s^{+'}, a). \end{aligned} \tag{2}$$

In the following section, we present DynaMITE-RL for learning a Bayes-optimal agent in a DLCMDP.

4. DynaMITE-RL

We detail DynaMITE-RL, first deriving a variational lower bound for learning a DLCMDP posterior model, then outlining three principals for training DLCMDPs, and finally integrating them into our training objective.

Variational Inference for Dynamic Latent Contexts. Given that we do not have direct access to the transition and reward functions of the DLCMDP, following Zintgraf et al. (2020), we infer the posterior $p(m \mid \tau_{:t})$, and reason about the latent embedding m instead. Since exact posterior computation over m is computationally infeasible, given the need to marginalize over task space, we introduce the

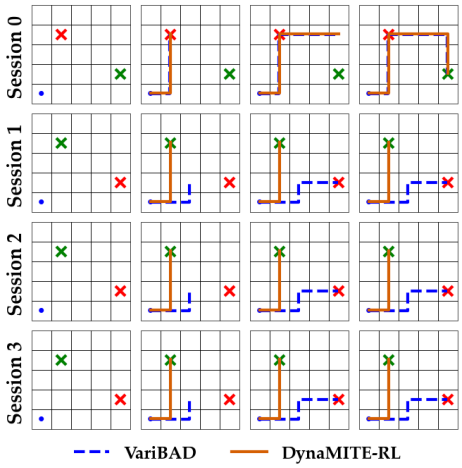

Figure 2. Qualitative behavior of one DLCMDP episode comparing trained VariBAD and DynaMITE-RL. VariBAD does not model the transition dynamics of the latent context and fails to adapt to the changing goal location. By contrast, DynaMITE-RL correctly infers the transition and consistently reaches the rewarding cell.

variational posterior $q_\phi(m \mid \tau_{:t})$, parameterized by $\phi \in \mathbb{R}^d$, to enable fast inference at every step. Our learning objective maximizes the log-likelihood $\mathbb{E}_\pi[\log p(\tau)]$ of observed trajectories. In general, the true posterior over the latent context is intractable, as is the empirical estimate of the log-likelihood. To circumvent this, we derive the *evidence lower bound (ELBO)* (Kingma & Welling, 2014) to approximate the posterior over m under the variational inference framework.

Let $\mathcal{Z} = \{m^i\}_{i=0}^{K-1}$ be the latent contexts for each of the K sessions in an episode (K is an a priori unknown random variable—we do not observe the number of sessions in an episode). As defined previously, Ω is the collection of the session terminations. We use a parametric generative distribution model for the state-reward trajectory, conditioned on the action sequence: $p_\theta(s_0, r_1, s_1, \ldots, r_H, s_H \mid a_0, \ldots, a_{H-1})$. In what follows, we drop the conditioning on $a_{:H-1}$ for brevity.

The variational lower bound can be expressed as:

$$\begin{aligned}\log p_\theta(\tau) &\geq \underbrace{\mathbb{E}_{q_\phi(\mathcal{Z},\Omega \mid \tau_{:t})}\big[\log p_\theta(\tau \mid \mathcal{Z}, \Omega)\big]}_{\text{reconstruction}} \\ &\quad - \underbrace{D_{KL}(q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})) \parallel p_\theta(\mathcal{Z}, \Omega))}_{\text{regularization}} \\ &= \mathcal{L}_{\text{ELBO},t},\end{aligned} \tag{3}$$

which can be estimated via Monte Carlo sampling over a learnable approximate posterior q_ϕ. In optimizing the reconstruction loss of session transitions and rewards, the learned latent variables should capture the unobserved MDP parameters. The full derivation of the ELBO for a DLCMDP is provided in Appendix A.

Figure 2 depicts a (qualitative) didactic GridWorld example with varying goals. The VariBAD agent does not account for latent goal dynamics and gets stuck after reaching the goal in the first session. By contrast, DynaMITE-RL employs the latent context dynamics model to capture goal changes, and adapts to the changes across sessions.

Consistency of Latent Information. In the DLCMDP formulation, each session is itself an MDP with a latent context fixed across the session. This within-context stationarity means new observations can only increase the information the agent has about this context. In other words, the agent's posterior over latent contexts gradually hone in on the true latent distribution. Although this true distribution remain unknown, this insight suggest the use of a *session-based consistency loss*, which penalizes an increase in KL-divergence between the current and final posterior belief within a session. Let $d_{H-1} = 1$ and $t_i \in \{0, \ldots, H\}$ be a random variable denoting the last timestep of session $i \in \{0, \ldots, K{-}1\}$, i.e., $t_i = \min\{t' \in \mathbb{Z}_{\geq 0} : \sum_{t=0}^{t'} d_t = i + 1\}$. At each time t in session i, we define this loss as

$$\begin{aligned}\mathcal{L}_{\text{consistency},t} = \\ \max\{&D_{KL}(q_\phi(m^i \mid \tau_{:t+1}) \parallel q_\phi(m^i \mid \tau_{:t_i})) \\ &- D_{KL}(q_\phi(m^i \mid \tau_{:t}) \parallel q_\phi(m^i \mid \tau_{:t_i})),\ 0\}\end{aligned} \tag{4}$$

where $q_\phi(m^i \mid \tau_{:t_i})$ is the final posterior in session i. Using temporal consistency to regularize inference introduces an explicit inductive bias that allows for better posterior estimation.

Remark 4.1. We introduce session-based consistency for DLCMDPs, though it is also relevant in single-session settings with non-dynamic latent context. Indeed, as we discuss below, while VariBAD focuses on single sessions, it does not constrain the latent's posterior to be identical to final posterior belief. Consistency may be useful in settings where the underlying latent variable is stationary, but may hurt performance when this variable is indeed changing. Since our modeling approach allows latent context changes across sessions, incorporating consistency regularization does not generally hurt performance.

Latent Belief Conditioning. Unlike the usual BAMDP framework, DLCMDPs allow one to model temporal changes of latent contexts via dynamics $T(m' \mid m)$ across sessions. To incorporate this model into belief estimation, in addition to the history $(\tau_{:t}, d_{:t})$, we condition the posterior on the final latent belief $q_\phi(m', d' \mid m, d, \tau_{:t})$ from the previous session, and impose KL-divergence matching between this belief and the prior distribution $p_\theta(m' \mid m)$.

Reconstruction Masking. When the agent is at time t, Zintgraf et al. (2020) encode past interactions to obtain

Algorithm 1 DynaMITE-RL

Input: env, policy, critic, encoder, decoder
for iter $= 1$ to N **do**
  Collect DLCMDP episode
  Train VAE by maximizing ELBO using Eq. (6)
  Train policy and critic with any online RL algorithm
end for

the current posterior $q_\phi(m \mid \tau_{:t})$ since this is all the information available for inference about the current task (see Eq. (3)). They use this posterior to decode the entire trajectory—*including future transitions*—from different sessions to optimize the lower bound during training. The insight is that decoding both the past and future allows the posterior model to perform inference about unseen states. However, we observe that when the latent context is stochastic, reconstruction over the full sequence is detrimental to training efficiency. The model is attempting to reconstruct transitions outside of the current session that may be irrelevant or biased given the latent-state dynamics, rendering it a more difficult learning problem. Instead we reconstruct only the transitions within the session defined by the termination indicators, i.e.,

$$\begin{aligned}\mathcal{L}_{\text{session-ELBO},t} = \quad & (5)\\ & \mathbb{E}_{q_\phi(\mathcal{Z},\Omega|\tau_{:t})}\big[\log p_\theta(\tau_{t_{i-1}+1:t_i} \mid \mathcal{Z},\Omega)\big]\\ & - D_{KL}(q_\phi(\mathcal{Z},\Omega \mid \tau_{:t})) \,\|\, p_\theta(\mathcal{Z},\Omega))\end{aligned}$$

if time t is in session i.

DynaMITE-RL. By incorporating the three modifications above, we obtain at the following training objective for our variational meta-RL approach:

$$\begin{aligned}\mathcal{L}_{\text{DynaMITE-RL}}(\theta,\phi) = \quad & (6)\\ & \sum_{t=0}^{H-1}\Big[\mathcal{L}_{\text{session-ELBO},t}(\theta,\phi) + \beta\mathcal{L}_{\text{consistency},t}(\phi)\Big]\end{aligned}$$

where β is a hyperparameter for the consistency loss. We present simplified pseudocode for training DynaMITE-RL in Algorithm 1 and a detailed algorithm in Appendix B.

Implementation Details. We use proximal policy optimization (PPO) (Schulman et al., 2017) for online RL training. We introduce a posterior inference network that outputs a Gaussian over the latent context for the i-th session and the session termination indicators, $q_\phi(m^i, d_{:t} \mid \tau_{:t}, m^{i-1})$, conditioned on the history and posterior belief from the previous session. We parameterize the inference network as an RNN, specifically a uni-directional gated recurrent unit (Cho et al., 2014), with shared parameters, but different multi-layer perceptron (MLP) output heads: one head for predicting the logits for session termination, and another for the posterior belief. In practice, the posterior MLP outputs the parameters

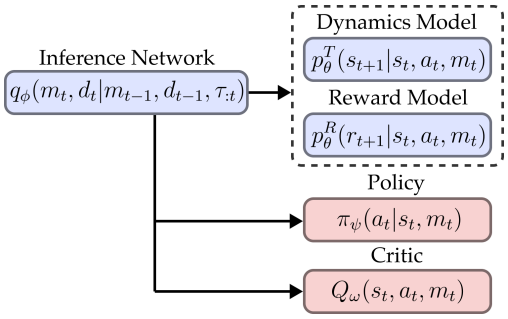

Figure 3. Model architecture of DynaMITE-RL.

of a Gaussian $q_{\phi_m}(m^i \mid \tau_{:t}, m^{i-1}) = \mathcal{N}(\mu(\tau_{:t}), \Sigma(\tau_{:t}))$. The session termination network applies a sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ to the MLP output. A decoder network, also parameterized using MLPs, reconstructs transitions and rewards given the session's latent context m^i, current state s_t, and action a_t, i.e., $p_\theta^T(s_{t+1} \mid s_t, a_t, m_t)$ and $p_\theta^R(r_{t+1} \mid s_t, a_t, m_t)$.

The final objective is to maximize:

$$\mathcal{L}(\theta,\phi,\psi) = \mathbb{E}\Big[\mathcal{J}_\pi(\psi) + \lambda\mathcal{L}_{\text{DynaMITE-RL}}(\phi,\theta)\Big] \qquad (7)$$

where $\mathcal{J}$ is the expected return and λ trades off this return with the variational inference objective. Following PPO (Schulman et al., 2017), the actor loss $\mathcal{J}_\pi$ and critic loss $\mathcal{J}_\omega$ are, respectively, $\mathcal{J}_\pi = \mathbb{E}_{\tau\sim\pi_\psi}[\log \pi_\theta(a \mid s, m)\hat{A}]$ and $\mathcal{J}_\omega = \mathbb{E}_{\tau\sim\pi_\psi}[(Q(s,a,m) - (r + V(s',m))^2]$ where V is the target network and $\hat{A}$ is the advantage function. We also add an entropy bonus to ensure sufficient exploration in more complex domains. Figure 3 depicts the implemented model architecture above.

5. Experiments

We present experiments that demonstrate, while VariBAD and other meta-RL methods struggle to learn good policies given nonstationary latent contexts, DynaMITE-RL exploits the causal structure of a DLCMDP to more efficiently learn performant policies. We compare our approach to several state-of-the-art meta-RL baselines, showing its significantly better performance.

Environments. We test DynaMITE-RL on a suite of tasks including gridworld navigation, continuous control, and human-in-the-loop robot assistance as shown in Figure 4. Several of these environments are commonly used in the meta-RL literature. For example, gridworld navigation and MuJoCo (Todorov et al., 2012) locomotion tasks are considered by Zintgraf et al. (2020) and Choshen & Tamar (2023). We modify these environments to incorporate temporal shifts in the reward and/or environment dynamics. To achieve good performance under these conditions, a learned

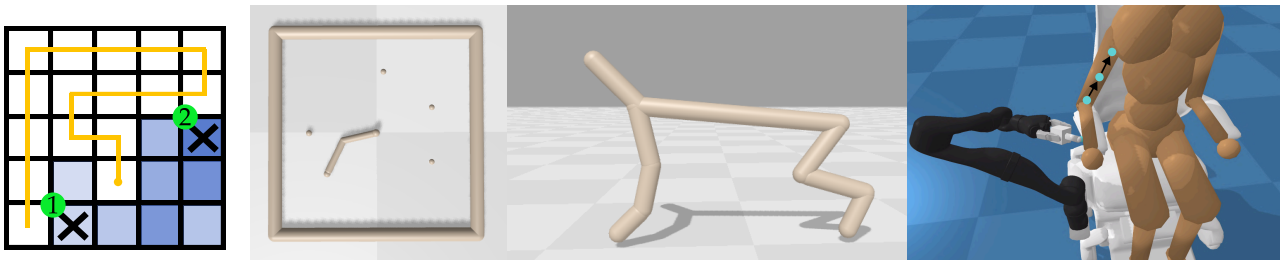

Figure 4. The environments considered in evaluating DynaMITE-RL. Each environment exhibits some change in reward and/or dynamics between sessions including changing goal locations (left and middle left), changing target velocities (middle right), and evolving user preferences of itch location (right).

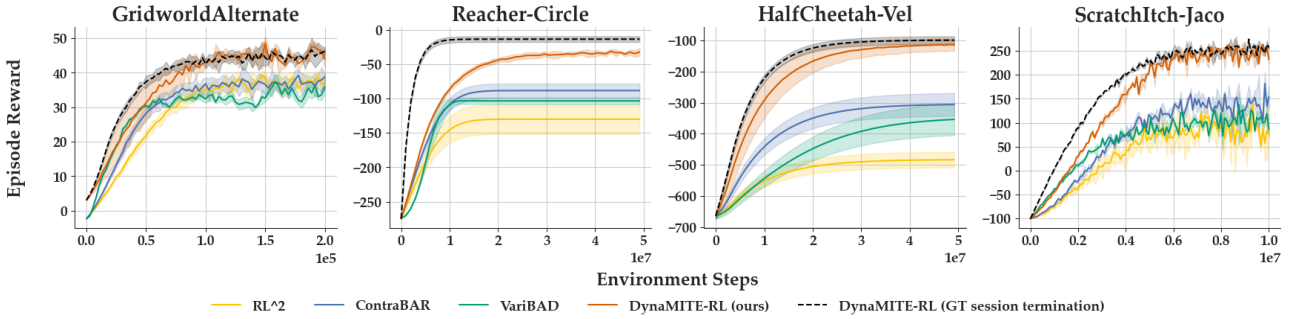

Figure 5. Learning curves for **DynaMITE-RL** and baseline methods. Shaded areas represent standard deviation over 5 different random seeds for each method and 3 for ScratchItch. In each of the evaluation environments, we observe that **DynaMITE-RL** exhibits better sample efficiency and converges to better environment returns than the baseline methods.

policy must adapt to the latent state dynamics. More details about the environments and hyperparameters can be found in Appendix D and E.

Gridworld. We modify the Gridworld environment used by Zintgraf et al. (2020). In a 5×5 gridworld, two possible goals are sampled uniformly at random in each episode. One of the two goals has a $+1$ reward while the other has 0 reward. The rewarding goal location changes after each session according to a predefined transition function. Goal locations are provided to the agent in the state—the only latent information is which goal has positive reward.

Continuous Control. We experiment with two tasks from OpenAI Gym (Brockman et al., 2016): Reacher and HalfCheetah. Reacher is a two-jointed robot arm and its task is to reach a 2D goal location that moves along a circular path according to some unknown transition function. HalfCheetah is a locomotion task which we modify to incorporate changing latent contexts w.r.t. the target direction (HalfCheetah-Dir), target velocity (HalfCheetah-Vel), and additionally magnitude of opposing wind forces on the agent (HalfCheetah-Wind+Vel). The results for HalfCheetah-Dir and HalfCheetah-Wind+Vel can be found in Appendix C.

Assistive Itch Scratching (Erickson et al., 2020). The environment consists of a human and a wheelchair-mounted 7-degree-of-freedom (DOF) Jaco robot arm. The human has limited-mobility and requires robot assistance to scratch an itch. We simulate stochastic latent context by moving the itch location—unobserved by the agent—along the human's right arm.

Baselines. We compare DynaMITE-RL to several state-of-the-art (approximately) Bayes-optimal meta-RL methods including RL^2 (Duan et al., 2016), VariBAD (Zintgraf et al., 2020), and ContraBAR (Choshen & Tamar, 2023). RL^2 (Duan et al., 2016) is an RNN-based policy gradient method which encodes environment transitions in the hidden state and maintains them across episodes. VariBAD reduces to RL^2 without the decoder and the variational reconstruction objective for environment transitions. ContraBAR employs a contrastive learning objective to discriminate future observations from negative samples to learn an *approximate* sufficient statistic of the history. As Zintgraf et al. (2020) already demonstrate better performance by VariBAD than posterior sampling methods (e.g., PEARL (Rakelly et al., 2019)) we exclude such methods from our comparison.

DynaMITE-RL outperforms prior meta-RL methods in a DLCMDP. In Figure 5, we show the learning curves for DynaMITE-RL and baseline methods. We first observe that **DynaMITE-RL** significantly outperforms the baselines

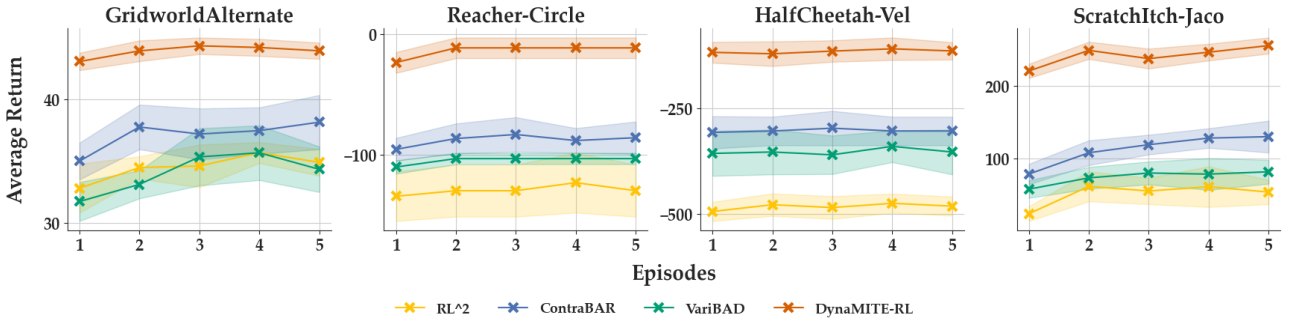

Figure 6. Average test-time performance on MuJoCo tasks and ScratchItch task, trained separately with 5 seeds for MuJoCo tasks and 3 for itching task. The meta-trained policies are rolled out for 5 episodes to show how they adapt to the task. The returns averaged across the task with 95% confidence intervals shaded. We demonstrate that in our DLCMDP setting, the baseline methods struggle to adapt to the changing dynamics of the environment while our method learns the latent transitions and achieves good performance across all domains.

across all domains in terms of sample efficiency and average environment returns. **RL**^{2}, **VariBAD**, and **ContraBAR** all perform poorly in the DLCMDP, converging to a suboptimal policy. By contrast, **DynaMITE-RL** accurately models the latent dynamics and consistently achieves high rewards despite the nonstationary latent context. We also evaluate an oracle with access to ground-truth session terminations and find that **DynaMITE-RL** with learned session terminations effectively recovers session boundaries and matches oracle performance with sufficient training. Following Zintgraf et al. (2020), we measure test-time performance of meta-trained policies by evaluating per-episode return for 5 consecutive episodes, see Figure 6. **DynaMITE-RL** and all of the baselines are designed to maximize reward *within a single rollout* hence they generally plateau after a single episode. Our empirical results validate that **DynaMITE-RL** learns a policy robust to changing latent contexts at inference time, while the baseline methods fail to adapt and get stuck in suboptimal behavior. The full set of evaluation results for all environments can be found in Appendix C.

Each component of DynaMITE-RL contributes to efficient learning in a DLCMDP: We ablate the three key components of **DynaMITE-RL** to understand their impact on the resulting policy. We compare full **DynaMITE-RL** to: (i) DynaMITE-RL w/o Consistency, which does not include consistency regularization; (ii) DynaMITE-RL w/o Conditioning, which does not include latent conditioning; and (iii) DynaMITE-RL w/o SessRecon, which does not include session reconstruction. In Figure 7, we report the learning curves for each of these ablations and vanilla VariBAD for reference. First, we observe that without prior latent belief conditioning, the model converges to a suboptimal policy slightly better than **VariBAD**, confirming the importance of modeling the latent transition dynamics of a DLCMDP. Second, we find that session consistency regularization reinforces the inductive bias of changing dynamics and improves the sample efficiency of learning an accurate posterior model in DLCMDPs. Finally, session reconstruction masking also improves the sample efficiency by not reconstructing terms that are irrelevant and potentially biased.

DynaMITE-RL is robust to varying levels of latent stochasticity. We study the effect of varying the number of sessions—in effect the number of latent context switches—over an episode of a fixed time horizon. For the HalfCheetah-Vel environment, we fix the episode horizon $H = 400$, distributing it across K sessions such that the lengths of the first $K-1$ sessions are sampled from a Poisson distribution, $l_i \sim \text{Poisson}(\frac{H}{K}) > 0$ and the final session has length $H - \sum_{i=1}^{K-1} l_i$.[1] As we increase the number of sessions, session length decreases and there will be more latent context switches. Setting the number of context switches to 1 is equivalent to a latent MDP episode with a static latent variable. As shown in Figure 8, **DynaMITE-RL** performs better, on average, than **VariBAD**, with lower variance in a latent MDP. We hypothesize that, in the case of latent MDP, consistency regularization helps learn a more accurate posterior model by enforcing the inductive bias that the latent is static. Otherwise, there is no inherent advantage in modeling the latent dynamics if it is stationary. As we gradually increase the number of context switches, the problem becomes more difficult and closer to a general POMDP. **VariBAD** performance decreases drastically because it is unable to model the changing latent dynamics while **DynaMITE-RL** is less affected, highlighting the robustness of our approach. When we set the number of contexts equal to the episode horizon length, we recreate a fully general POMDP and again the performance between **VariBAD** and **DynaMITE-RL** converges.

[1] We resample if the length of any session is not positive.

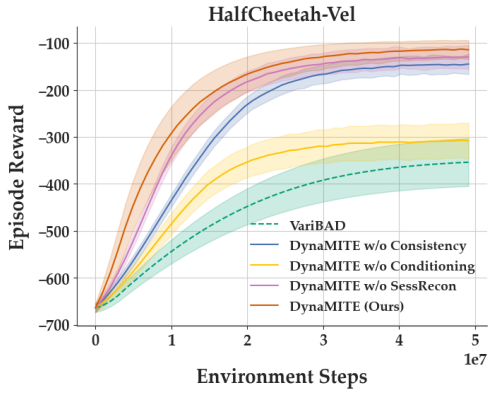

Figure 7. Ablating components of **DynaMITE-RL**. We observe that prior latent conditioning is crucial in achieving good performance in a DLCMDP. Additionally, consistency regularization and session reconstruction improve the sample efficiency and convergence to a better performing policy.

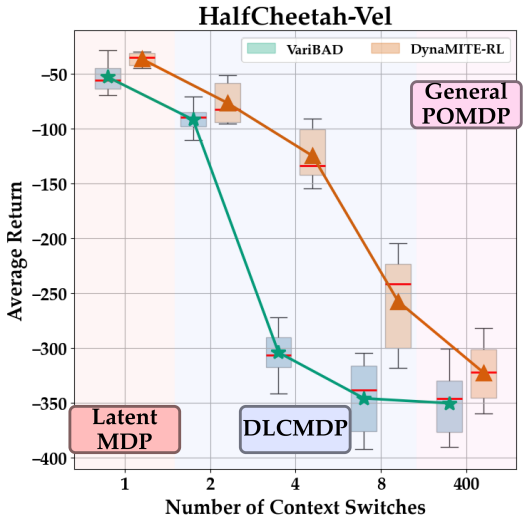

Figure 8. Ablation studying the effect of number of latent context switches in an episode with fixed horizon on **VariBAD** and **DynaMITE-RL** in HalfCheetah-Vel. The boxplot shows the distribution over evaluation returns for 25 rollouts of trained policies. The rollout length is fixed to be 400 timesteps divided across sessions. When the number of context switches is 1 we have a latent MDP and when it is 400 this is equivalent to a general POMDP.

6. Related Work

POMDPs provide a general framework modeling non-stationality and partial observability in sequential decision problems. Many model variants have been introduced, defining a rich spectrum between episodic MDPs and POMDPs. The Bayes-adaptive MDP (BAMDP) (Duff, 2002) and the hidden parameter MDP (HiP-MDP) (Killian et al., 2017) are special cases of POMDPs in which environment parameters are unknown and the goal is to infer these parameters online during an episode. However, neither framework addresses the dynamics of the latent parameters across sessions, but instead assume the latent context is constant throughout the episode. By contrast, DLCMDPs allow one to model the dynamics of the latent state, allowing better posterior updates at inference time.

DynaMITE-RL shares conceptual similarities with other meta-RL algorithms. Firstly, optimization-based techniques (Finn et al., 2017; Clavera et al., 2018; Rothfuss et al., 2018) learn neural network policies that can quickly adapt to new tasks at test time using policy gradient updates. However, these methods do not optimize for Bayes-optimal behavior and generally exhibit suboptimal test-time adaptation. Context-based meta-RL techniques aim to learn policies that directly infer task parameters at test time, conditioning the policy on the posterior belief. Such methods include recurrent memory-based architectures (Duan et al., 2016; Wang et al., 2016; Lee et al., 2018) and variational approaches (Humplik et al., 2019; Zintgraf et al., 2020; Dorfman et al., 2021). VariBAD, closest to our work, uses variational inference to approximate Bayes-optimal policies. However, we have demonstrated above the limitations of VariBAD in DLCMDPs, and have developed several crucial modifications to drive effective learning a highly performant policies in our setting.

7. Conclusion

We have developed DynaMITE-RL, a meta-RL method to approximate Bayes-optimal behavior using a latent variable model. We presented the dynamic latent contextual Markov Decision Process (DLCMDP), a model in which latent context information changes according to an unknown transition function, that captures many natural settings. We derived a graphical model for this problem setting and formalized it as an instance of a POMDP. DynaMITE-RL is designed to exploit the causal structure of this model, and in a didactic GridWorld environment and several challenging continuous control tasks, we demonstrated that it outperforms existing meta-RL methods w.r.t. both learning efficiency and test-time adaptation.

There are a number of directions for future research. While we only consider Markovian latent dynamics here (i.e. future latent states are independent of prior latent states given the current latent state), we plan to investigate richer non-Markovian latent dynamics. In real-world applications like RSs, in which an agent interacts with users over long periods of time, an RNN architecture may not have sufficient capacity to capture long histories of user interaction. It would be interesting to explore transformers to model long interaction histories in addition to complex non-Markovian latent dynamics.

8. Broader Impacts

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Beck, J., Vuorio, R., Liu, E. Z., Xiong, Z., Zintgraf, L. M., Finn, C., and Whiteson, S. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

Bellman, R. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679—-84, 1957.

Bertsekas, D. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

Biyik, E., Margoliash, J., Alimo, S. R., and Sadigh, D. Efficient and safe exploration in deterministic markov decision processes with unknown transition models. In *American Control Conference*, pp. 1792–1799. IEEE, 2019.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Cao, Z., Bıyık, E., Wang, W. Z., Raventos, A., Gaidon, A., Rosman, G., and Sadigh, D. Reinforcement learning based control of imitative policies for near-accident driving. *Robotics: Science and Systems*, 2020.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014.

Choshen, E. and Tamar, A. Contrabar: Contrastive bayes-adaptive deep rl. In *International Conference on Machine Learning*, volume 202, pp. 6005–6027, 2023.

Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., and Abbeel, P. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pp. 617–629. PMLR, 2018.

Dorfman, R., Shenfeld, I., and Tamar, A. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Neural Information Processing Systems*, 34:4607–4618, 2021.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Duff, M. O. *Optimal learning: computational procedures for bayes-adaptive markov decision processes*. PhD thesis, University of Massachusetts Amherst, 2002.

Erickson, Z., Gangaram, V., Kapusta, A., Liu, C. K., and Kemp, C. C. Assistive gym: A physics simulation framework for assistive robotics. In *IEEE International Conference on Robotics and Automation*. IEEE, 2020.

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

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483, 2015.

Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., and Wang, W. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Humplik, J., Galashov, A., Hasenclever, L., Ortega, P. A., Teh, Y. W., and Heess, N. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

Ie, E., Hsu, C., Mladenov, M., Jain, V., Narvekar, S., Wang, J., Wu, R., and Boutilier, C. RecSim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847*, 2019.

Jannach, D., Manzoor, A., Cai, W., and Chen, L. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

Jawaheer, G., Weller, P., and Kostkova, P. Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Transactions on Interactive Intelligent Systems*, 4(2):1–26, 2014.

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

Killian, T. W., Daulton, S., Konidaris, G., and Doshi-Velez, F. Robust and efficient transfer learning with hidden parameter markov decision processes. *Neural Information Processing Systems*, 2017.

Kim, C., Park, J., Shin, J., Lee, H., Abbeel, P., and Lee, K. Preference transformer: Modeling human preferences using transformers for rl. *International Conference of Learning Representations*, 2023.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Kwon, J., Efroni, Y., Caramanis, C., and Mannor, S. Rl for latent mdps: Regret guarantees and a lower bound. *Neural Information Processing Systems*, 34:24523–24534, 2021.

Lee, G., Hou, B., Mandalika, A., Lee, J., Choudhury, S., and Srinivasa, S. S. Bayesian policy optimization for model uncertainty. *International Conference on Learning Representations*, 2018.

Liu, S., See, K. C., Ngiam, K. Y., Celi, L. A., Sun, X., and Feng, M. Reinforcement learning for clinical decision support in critical care: comprehensive review. *Journal of Medical Internet Research*, 22(7):e18477, 2020.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

Ross, S., Chaib-draa, B., and Pineau, J. Bayes-adaptive pomdps. *Neural Information Processing Systems*, 2007.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search. *International Conference on Learning Representations*, 2018.

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

Steimle, L. N., Kaufman, D. L., and Denton, B. T. Multi-model markov decision processes. *IISE Transactions*, 53 (10):1124–1139, 2021.

Tennenholtz, G., Hallak, A., Dalal, G., Mannor, S., Chechik, G., and Shalit, U. On covariate shift of latent confounders in imitation and reinforcement learning. *International Conference of Learning Representations*, 2022.

Tennenholtz, G., Merlis, N., Shani, L., Mladenov, M., and Boutilier, C. Reinforcement learning with history dependent dynamic contexts. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 34011–34053. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/tennenholtz23a.html.

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

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

Yu, C., Liu, J., Nemati, S., and Yin, G. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. VariBAD: A very good method for bayes-adaptive deep rl via meta-learning. *International Conference of Learning Representations*, 2020.

A. Full ELBO Derivation for DLCMDP

We will define a full trajectory $\tau = \{s_0, a_0, r_1, s_1, a_1, \ldots, r_{H-1}, s_H\}$ where H is the horizon. $\tau_{:t}$ is the history of interactions up to a global timestep t, i.e. $\tau_{:t} = \{s_0, a_0, r_1, s_1, a_1, \ldots r_{t-1}, s_t\}$.

Let $\mathcal{Z} = \{m^0, \ldots, m^{K-1}\}$ be the collection of latent contexts in a trajectory where K is a random variable representing the number of switches the latent variable will have until time H, i.e., $K = \sum_{t=0}^{H-1} d_t$. Additionally, we denote d_t as the session termination prediction at timestep t but $d_{H-1} \equiv 1$.

We divide a full trajectory into sessions and define a discrete random variable $t_i \in \{0, \ldots, H-1\}$ be a random variable denoting the last timestep of session $i \in \{0, \ldots, K-1\}$, i.e., $t_i = \min\{t' \in \mathbb{Z}_{\geq 0} : \sum_{t=0}^{t'} d_t = i+1\}$, with $t_{-1} \equiv -1$. We also denote the next session index $i' = i+1$.

An arbitrary session i' can then be represented as, $\{s_{t_i+1}, a_{t_i+1}, r_{t_i+1}, s_{t_i+2}, \ldots, s_{t_{i'}-1}, a_{t_{i'}-1}, r_{t_{i'}}\}$.

At any time-step t, we want to maximize the log-likelihood of the full dataset of trajectories, $\mathcal{D}$, collected following policy π, e.g. $\mathbb{E}_\pi[\log p_\theta(\tau)]$. However, with the presence of latent variables, whose samples cannot be observed in the training data, estimating the empirical log-likelihood is generally intractable. Instead, we optimize for the evidence lower bound (ELBO) of this function with a learned approximate posterior, q_ϕ.

We then define the posterior inference model, $q_\phi(\mathcal{Z}, d_{:H} \mid \tau_{:t})$, which outputs the posterior distribution for the latent context and session termination predictions conditioned on the trajectory history up until timestep t.

Below we provide the derivation for the variational lower bound of the log-likelihood function $\log p_\theta(\tau)$ for a single trajectory:

$$
\begin{aligned}
\log p_\theta(\tau) &= \log \int_{\mathcal{Z},\Omega} p_\theta(\tau, \mathcal{Z}, \Omega) \\
&= \log \int_{\mathcal{Z},\Omega} p_\theta(\tau, \mathcal{Z}, \Omega) \frac{q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})}{q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})} \\
&= \log \mathbb{E}_{q_\phi(\mathcal{Z},\Omega|\tau_{:t})}\left[\frac{p_\theta(\tau, \mathcal{Z}, \Omega)}{q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})}\right] \\
&= \log \mathbb{E}_{q_\phi(\mathcal{Z},\Omega|\tau_{:t})}\left[\frac{p_\theta(\tau \mid \mathcal{Z}, \Omega)\, p_\theta(\mathcal{Z}, \Omega)}{q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})}\right] \\
&\geq \mathbb{E}_{q_\phi(\mathcal{Z},\Omega|\tau_{:t})}\big[\log p_\theta(\tau \mid \mathcal{Z}, \Omega) + \log p_\theta(\mathcal{Z}, \Omega) - \log(q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t}))\big] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathcal{Z},\Omega|\tau_{:t})}\big[\log p_\theta(\tau \mid \mathcal{Z}, \Omega)\big]}_{\text{reconstruction}} - \underbrace{D_{KL}(q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})) \,||\, p_\theta(\mathcal{Z}, \Omega))}_{\text{regularization}} \\
&= \text{ELBO}_t(\theta, \phi)
\end{aligned}
$$

We extend this to derive the lower bound for all trajectories in dataset $\mathcal{D}$.

$$
\mathbb{E}_{\tau\sim\mathcal{D}}\Big[\log p_\theta(\tau)\Big] = \mathbb{E}_{\tau\sim\mathcal{D}}\Big[\mathbb{E}_{q_\phi(\mathcal{Z},\Omega|\tau_{:t})}\big[\log p_\theta(\tau \mid \mathcal{Z}, \Omega)\big] - D_{KL}(q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})) \,||\, p_\theta(\mathcal{Z}, \Omega))\Big]
$$

Prior:

$$
p_\theta(\mathcal{Z}, \Omega) = p_\theta(m^0 \mid d_{:t_0}) p_\theta(d_{:t_0}) \prod_{i=0}^{K-2} p_\theta(m^{i'} \mid m^i, d_{t_i+1:t_{i'}}) p_\theta(d_{t_i+1:t_{i'}})
$$

Variational Posterior:

$$q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t}) = q_\phi(m^0 \mid \tau_{:t_0}, d_{:t_0}) q_\phi(d_{:t_0}) \prod_{i=-1}^{K-2} q_\phi(m^{i'} \mid \tau_{t_i+1:t_{i'}}, m^i, d_{t_i+1:t_{i'}}) q_\phi(d_{t_i+1:t_{i'}})$$

Reconstruction Term:

$$\begin{aligned}
\log p_\theta(\tau \mid \mathcal{Z}, \Omega) &= \log p_\theta(s_0, r_1, \ldots, r_{H-1}, s_H \mid \mathcal{Z}, \Omega, a_{:H-1}) \\
&= \log \prod_{i=-1}^{K-2} p_\theta(\{(s_t, r_t)\}_{t=t_i+1}^{t_{i'}} \mid \mathcal{Z}, \Omega, a_{:H-1}) \\
&= \log \prod_{i=-1}^{K-2} \left[p_\theta(s_{t_i+1}) \prod_{t=t_i+1}^{t_{i'}} p_\theta(s_{t+1} \mid s_t, a_t, \mathcal{Z}, d_t)\, p_\theta(r_{t+1} \mid s_t, a_t, \mathcal{Z}, d_t) \right] \\
&= \sum_{i=-1}^{K-2} \left[\log p_\theta(s_{t_i+1}) + \sum_{t=t_i+1}^{t_{i'}} \log p_\theta(s_{t+1}, r_{t+1} \mid s_t, a_t, \mathcal{Z}, d_t) \right]
\end{aligned}$$

Putting it all together:

$$\begin{aligned}
\log p_\theta(\tau) &\geq \underbrace{\mathbb{E}_{q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})}\big[\log p_\theta(\tau \mid \mathcal{Z}, \Omega)\big]}_{\text{reconstruction}} - \underbrace{D_{KL}(q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})) \,||\, p_\theta(\mathcal{Z}, \Omega))}_{\text{regularization}} \\
&= \mathbb{E}_{q_\phi(\mathcal{Z}, \Omega \mid \tau_{:t})}\{ \sum_{i=-1}^{K-2} \left[\log p_\theta(s_{t_i+1} \mid \mathcal{Z}, d_{t_i}) + \sum_{t=t_i+1}^{t_{i'}} \log p_\theta(s_{t+1}, r_{t+1} \mid s_t, a_t, \mathcal{Z}, d_t) \right] \} \\
&- D_{KL}(q_\phi(m^0 \mid \tau_{:t_0}, d_{:t_0}) \parallel p_\theta(m^0 \mid d_{:H}) \\
&- \sum_{i=0}^{K-2} D_{KL}(q_\phi(m^{i'} \mid \tau_{t_i+1:t_{i'}}, m^i, d_{t_i+1:t_{i'}}) \parallel p_\theta(m^{i'} \mid m^i, d_{t_i+1:t_{i'}})) \\
&- \sum_{i=0}^{K-2} D_{KL}(q_\phi(d_{t_i+1:t_{i'}}) \parallel p_\theta(d_{t_i+1:t_{i'}}))
\end{aligned}$$

B. Pseudocode for DynaMITE-RL

Here we provide the pseudocode for training DynaMITE-RL and for rolling out the policy during inference time.

Algorithm 2 DynaMITE-RL

Input: env, $\alpha_\psi, \alpha_\omega$
Randomly initialize policy $\pi_\psi(a \mid s, m)$, critic $Q_\omega(s, a, m)$ decoder $p_\theta(s', r' \mid s, a, m)$, encoder $q_\phi(m'|\cdot)$, and replay buffer $\mathcal{D} = \emptyset$
for $i = 1$ to N **do**
  $\mathcal{D}[i] \leftarrow$ `COLLECT_TRAJECTORY`$(\pi_\psi, q_\phi, \text{env})$
  ▷ Train VAE
  Sample batches of trajectories from $\mathcal{D}$
  Compute ELBO with Eq. 6 and update p_θ, q_ϕ
  ▷ Update actor and critic using PPO
  $\psi \leftarrow \psi - \alpha_\psi \nabla_\psi \mathcal{J}_\pi$
  $\omega \leftarrow \omega - \alpha_\omega \nabla_\omega \mathcal{J}_\mathcal{Q}$
end for

Algorithm 3 COLLECT_TRAJECTORY

Input: π_θ, q_ϕ, env
$(s_0, m_0) \sim \nu_0$ {sample initial state and belief}
$k = 0$ {session index}
for $t = 0$ to $H - 1$ **do**
  $a_t \sim \pi_\psi(a_t \mid s_t, m_t)$ {get action}
  $(s_{t+1}, r_{t+1}) =$ env.step(a_t) {env step}
  ▷ Posterior update
  if $k == 0$ **then**
    $m_{t+1}, d_{t+1} \sim q_\phi(\cdot \mid \tau_{:t+1})$
  else
    $m_{t+1}, d_{t+1} \sim q_\phi(\cdot \mid \tau_{:t+1}, m_{t_{k-1}})$
  end if
  if session-terminate(d_{t+1}) **then**
    k += 1 {increment session index}
    $(s_{t+1}, m_{t+1}) \sim \nu_0$ {reset the state}
  end if
end for

C. Additional Experimental Results

Here, we provided the full set of unnormalized experimental results.

	RL^2	ContraBAR	VariBAD	DynaMITE-RL
Gridworld	33.4 ± 1.6	34.5 ± 0.9	31.8 ± 1.9	$\mathbf{42.9} \pm 0.5$
Reacher	-150.6 ± 1.2	-101.6 ± 3.2	-102.4 ± 4.2	$\mathbf{-8.4} \pm 5.1$
HalfCheetah-Dir	-420 ± 4.6	-256.5 ± 3.2	-242.5 ± 5.6	$\mathbf{-68.5} \pm 2.3$
HalfCheetah-Vel	-513.2 ± 8.7	-312.3 ± 4.8	-363.5 ± 3.2	$\mathbf{-146.0} \pm 8.1$
HalfCheetah-Wind+Vel	-493.5 ± 1.8	-243.4 ± 2.6	-188.5 ± 4.4	$\mathbf{-42.8} \pm 6.9$
ScratchItch	50.4 ± 16.8	114.6 ± 24.4	81.8 ± 6.9	$\mathbf{231.2} \pm 23.3$

Table 1. Average single episode returns for DynaMITE-RL and other benchmark algorithms across all environments. Results for all environments is averaged across 5 seeds beside ScratchItch which has 3 seeds. Algorithm with the highest average return are shown in bold. DynaMITE-RL achieves the highest return on all of the evaluation environments.

D. Evaluation Environment Description

In this section, we describe the details of the domains we used for our experiments. We provide visualizations of each simulation environment in Figure 4.

D.1. Gridworld Navigation with Alternating Goals

Following (Zintgraf et al., 2020), we extend the 5×5 gridworld environment as shown in Figure 2. For each episode, two goal locations are selected randomly. However, only one of the goal locations provide a positive reward when the agent arrives at the location. The rewarding goal location changes between sessions according to some transition dynamics. In our experiments, we simulate latent dynamics using a simple transition matrix: $\begin{bmatrix} 0.2 & 0.8 \\ 0.8 & 0.2 \end{bmatrix}$. Between each session, the goal location has a 20% chance of remaining the same as the previous session and 80% chance of switching to the other location. The agent receives a reward of -0.1 on non-goal cells and +1 at the goal cell, e.g.

$$r_t = \begin{cases} 1 & \text{if } s_t = g \\ -0.1 & \text{otherwise} \end{cases}$$

where s_t is the current state and g is the current rewarding goal cell. Similar to (Zintgraf et al., 2020), we set the horizon length to 15 and train on episodes with 4 sessions.

D.2. MuJoCo Continuous Control

For our study, we use the Brax (Freeman et al., 2021) simulator, a physics engine for large scale rigid body simulation written in JAX. We use JAX [2], a machine learning framework which has just-in-time (jit) compilation that perform operations on GPU and TPU for faster training and can optimize the execution significantly. We evaluate the capacity of our method to perform continuous control tasks with high-dimensional observation spaces and action spaces.

D.2.1. REACHER

Reacher is a two-joint robot arm task part of OpenAI's MuJoCo tasks (Brockman et al., 2016). The goal is to move the robot's end effector to a target 2D location. The goal locations change between each session following a circular path defined by: $[x, y] = [rcos(\alpha \cdot i), rsin(\alpha \cdot i)]$ where i is the session index, $\alpha \sim \mathcal{U}(0, 2\pi)$ is the initial angle, and $r \sim \mathcal{U}(0.1, 0.2)$ is the circle's radius. The observation space is 11 dimensional consisting of information about the joint locations and angular velocity. We remove the target location from the observation space. The action space is 2 dimension representing the torques applied at the hinge joints. The reward at each timestep is based on the distance from the reacher's fingertip to the target: $r_t = -||s_f - s_g||_2 - 0.05 \cdot ||a_t||_2$ where s_f is the (x, y) location of the fingertip and s_g for the target location.

D.2.2. HALF-CHEETAH

This environment builds off of the Half-Cheetah environment from OpenAI gym (Brockman et al., 2016), a MuJoCo locomotion task. In these tasks, the challenge is to move legged robots by applying torques to their joints via actuators. The state space is 17-dimensional, position and velocity of each joint. The initial state for each joint is randomized. The action space is a 6-dimensional continuous space corresponding to the torque applied to each of the six joints.

Half-Cheetah Dir(ection): In this environment, the agent has to run either forward or backward and this varies between session following a transition function. At the first session, the task is decided with equal probability. The reward is dependent on the goal direction:

$$r_t = \begin{cases} v_t + 0.5 \cdot ||a_t||_2 & \text{if task = forward} \\ -v_t + 0.5 \cdot ||a_t||_2 & \text{otherwise} \end{cases}$$

where v_t is the current velocity of the agent.

Half-Cheetah Vel(ocity): In this environment, the agent has to run forward at a target velocity, which varies between sessions. The task reward is: $r_t = -||v_s - v_g||_2 - 0.05 \cdot ||a_t||_2$, where v_s is the current velocity of the agent and v_g is the target velocity. The second term penalizes the agent for taking large actions. The target velocity varies between session according to: $v_g = 1.5 + 1.5\text{sin}(0.2 \cdot i)$.

Half-Cheetah Wind + Vel: The agent is additionally subjected to wind forces which is applied to the agent along the x-axis. Every time the agent takes a step, it drifts by the wind vector. The force is changing between sessions according to: $f_w = 10 + 10 \text{ sin}(0.3 \cdot i)$.

D.3. Assistive Gym

Our assistive itch scratching task is adapted from Assistive Gym (Erickson et al., 2020), similar to (Tennenholtz et al., 2022). Assistive Gym is a simulation environment for commercially available robots to perform 6 basic activities of daily living (ADL) tasks - itch scratching, bed bathing, feeding, drinking, dressing, and arm manipulation. We extend the itch scratching task in Assistive Gym.

The itch scratching task contains a human and a wheelchair-mounted 7-DOF Jaco robot arm. The robot holds a small scratching tool which it uses to reach a randomly target scratching location along the human's right arm. The target location gradually changes along the right arm according to a predefined function, $x = 0.5 + sin(0.2 \cdot i)$ where x is then projected onto a 3D point along the arm. Actions for each robot's 7-DOF arm are represented as changes in joint positions, $\mathbb{R}^7$. The observations include, the 3D position and orientation of the robot's end effector, the 7D joint positions of the robot's arm,

forces applied at the robot's end effector, and 3D positions of task relevant joints along the human body. Again, the target itch location is unobserved to the agent.

The robot is rewarded for moving its end effector closer to the target and applying less than 10 N of force near the target. Assistive Gym considers a person's preferences when receiving care from a robot. For example, a person may prefer the robot to perform slow actions or apply less force on certain regions of the body. Assistive Gym computes a human preference reward, $r_H(s)$, based on how well the robot satisfies the human's preferences at state s. The human preference reward is combined with the robot's task success reward $r_R(s)$ to form a dense reward at each timestep, $r(s) = r_R(s) + r_H(s)$.

The full human preference reward is defined as:

$$r_H(s) = -\alpha \cdot \omega[C_v(s), C_f(s), C_{hf}(s), C_{fd}(s), C_{fdv}(s), C_d(s), C_p(s)]$$

where α is a vector of activations in $\{0, 1\}$ depicting which components of the preference are used and ω is a vector of weights for each preference category. $C_\bullet(s)$ is the cost for deviating from the human's preference.

$C_v(s)$ for high end effector velocities. $C_f(s)$ for applying force away from the target location. $C_{hf}(s)$ for applying high forces near the target (> 10 N). $C_{fd}(s)$ for spilling food or water. $C_{fdv}(s)$ for food / water entering mouth at high velocities. $C_d(s)$ for fabric garments applying force to the body. $C_p(s)$ for applying high pressure with large tools.

For our itch-scratching task, we set $\alpha = [1, 1, 1, 0, 0, 0, 0]$ and $\omega = [0.25, 0.01, 0.05, 0, 0, 0, 0]$.

E. Implementation Details and Training Hyperparameters

In this section, we provide the hyperparameter values used for training each of the baselines and DynaMITE-RL. We also provide more detailed explanation of the model architecture used for each method.

We used Proximal Policy Optimization (PPO) training. The details of important hyperparameters use to produce the experimental results are presented in Table 2.

	Gridworld	Reacher	HalfCheetah	ScratchItch
Max episode length	60	400	400	200
Number of parallel processes	16	2048	2048	32
Value loss coefficient	0.5	-	-	-
Entropy coefficient	0.01	0.05	0.05	0.1
Learning rate	3e-4	-	-	-
Discount factor (γ)	0.99	-	-	-
GAE lambda (λ_{GAE})	0.95	-	-	-
Max grad norm	0.5	-	-	-
PPO clipping epsilon	0.2	-	-	-
Latent embedding dimension	5	16	16	16
ELBO loss coefficient	1.0	-	-	-
Policy learning rate	3e-4	-	-	-
VAE learning rate	3e-4	-	-	-
State/action/reward FC embed size	8	32	32	32
Consistency loss weight (β)	0.5	-	-	-
Variational loss weight (λ)	0.01	-	-	

Table 2. Training hyperparameters. Dashed entries means the same value is used across all environments.

We also employ several PPO training tricks detailed in (Huang et al., 2022), specifically normalizing advantage computation, using Adam epsilon $1e-8$, clipping the value loss, adding entropy bonus for better exploration, and using separate MLP networks for policy and value functions.

We use the same hyperparameters as above for RL^2 and VariBAD if applicable. For RL^2, the state and reward are embedded through fully connected (FC) layers, concatenated, and then passed to a GRU. The output is fed through another FC layer and then the network outputs the actions.

ContraBAR: Code based on the author's original implementation: https://github.com/ec2604/ContraBAR. ContraBAR uses contrastive learning, specifically Contrastive Predictive Coding (CPC) (Oord et al., 2018), to learn an information state representation of the history. They use CPC to discriminate between positive future observations o^+_{t+k} and K negative observations $\{o^-_{t+k}\}^K_{i=1}$ given the latent context c_t. The latent context is generated by encoding a sequence of observations through an autoregressive model. They apply an InfoNCE loss to train the latent representation.

DynaMITE-RL: The VAE architecture consists of a recurrent encoder, which at each timestep t takes as input the tuple (a_{t-1}, r_t, s_t). The state, action, and reward are each passed through a different linear layers followed by ReLU activations to produce separate embedding vectors. The embedding outputs are concatenated, inputted through an MLP with 2 fully-connected layers of size 64, and then passed to a GRU to produce the hidden state. Fully-connected linear output layers generate the parameters of a Gaussian distribution: $(\mu(\tau_{:t}), \Sigma(\tau_{:t}))$ for the latent embedding m. Another fully-connected layer produces the logit for the session termination. The reward and state decoders are MLPs with 2 fully-connected layers of size 32 with ReLU activations. They are trained my minimizing a Mean Squared Error loss against the ground truth rewards and states. The policy and critic networks are MLPs with 2 fully-connected layers of size 128 with ReLU activations. For the domains where the reward function is changing between sessions, we only train the reward-decoder. For HalfCheetah Wind + Vel, we also train the transition decoder.