Lifelong Learning with a Changing Action Set

Yash Chandak[1] **Georgios Theocharous**[2] **Chris Nota**[1] **Philip S. Thomas**[1]
[1]University of Massachusetts Amherst, [2]Adobe Research
{ychandak,cnota,pthomas}@cs.umass.edu theochar@adobe.com

Abstract

In many real-world sequential decision making problems, the number of available actions (decisions) can vary over time. While problems like catastrophic forgetting, changing transition dynamics, changing rewards functions, etc. have been well-studied in the lifelong learning literature, the setting where the size of the action set changes remains unaddressed. In this paper, we present first steps towards developing an algorithm that autonomously adapts to an action set whose size changes over time. To tackle this open problem, we break it into two problems that can be solved iteratively: inferring the underlying, unknown, structure in the space of actions and optimizing a policy that leverages this structure. We demonstrate the efficiency of this approach on large-scale real-world lifelong learning problems.

Introduction

Real-world problems are often non-stationary. That is, parts of the problem specification change over time. We desire autonomous systems that continually adapt by capturing the regularities in such changes, without the need to learn from scratch after every change. In this work, we address one form of *lifelong learning* for sequential decision making problems, wherein the set of possible actions (decisions) varies over time. Such a situation is omnipresent in real-world problems. For example, in robotics it is natural to add control components over the lifetime of a robot to enhance its ability to interact with the environment. In hierarchical reinforcement learning, an agent can create new *options* (Sutton, Precup, and Singh 1999) over its lifetime, which are in essence new actions. In medical decision support systems for drug prescription, new procedures and medications are continually discovered. In product recommender systems, new products are constantly added to the stock, and in tutorial recommendation systems, new tutorials are regularly developed, thereby continuously increasing the number of available actions for a recommender engine. These examples capture the broad idea that, for an agent that is deployed in real world settings, the possible decisions it can make changes over time, and motivates the question that we aim to answer: *how do we develop algorithms that can continually adapt to such changes in the action set over the agent's lifetime?*

Reinforcement learning (RL) has emerged as a successful class of methods for solving sequential decision making problems. However, excluding notable exceptions that we discuss later (Boutilier et al. 2018; Mandel et al. 2017), its applications have been limited to settings where the set of actions is fixed. This is likely because RL algorithms are designed to solve a mathematical formalization of decision problems called *Markov decision processes* (MDPs) (Puterman 2014), wherein the set of available actions is fixed. To begin addressing our lifelong learning problem, we first extend the standard MDP formulation to incorporate this aspect of changing action set size. Motivated by the regularities in real-world problems, we consider an underlying, unknown, structure in the space of actions from which new actions are generated. We then theoretically analyze the difference between what an algorithm can achieve with only the actions that are available at one point in time, and the best that the algorithm could achieve if it had access to the entire underlying space of actions (and knew the structure of this space). Leveraging insights from this theoretical analysis, we then study how the structure of the underlying action space can be recovered from interactions with the environment, and how algorithms can be developed to use this structure to facilitate lifelong learning.

As in the standard RL setting, when facing a changing action set, the parameterization of the policy plays an important role. The key consideration here is how to parameterize the policy and adapt its parameters when the set of available actions changes. To address this problem, we leverage the structure in the underlying action space to parameterize the policy such that it is invariant to the cardinality of the action set—changing the number of available actions does not require changes to the number of parameters or the structure of the policy. Leveraging the structure of the underlying action space also improves generalization by allowing the agent to infer the outcomes of actions similar to actions already taken. These advantages make our approach ideal for lifelong learning problems where the action set changes over time, and where quick adaptation to these changes, via generalization of prior knowledge about the impact of actions, is beneficial.

Related Works

Lifelong learning is a well studied problem (Thrun 1998; Ruvolo and Eaton 2013; Silver, Yang, and Li 2013; Chen and Liu 2016). Predominantly, prior methods aim to address catastrophic forgetting problems in order to leverage prior knowledge for new tasks (French 1999; Kirkpatrick et al. 2017; Lopez-Paz and others 2017; Zenke, Poole, and Ganguli 2017). Several meta-reinforcement-learning methods aim at addressing the problem of transfer learning, few-shot shot adaption to new tasks after training on a distribution of similar tasks, and automated hyper-parameter tuning (Xu, van Hasselt, and Silver 2018; Gupta et al. 2018; Wang et al. 2017; Duan et al. 2016; Finn, Abbeel, and Levine 2017). Alternatively, many lifelong RL methods consider learning online in the presence of *continuously* changing transition dynamics or reward functions (Neu 2013; Gajane, Ortner, and Auer 2018). In our work, we look at a complementary aspect of the lifelong learning problem, wherein the size of the action set available to the agent change over its lifetime.

Our work also draws inspiration from recent works which leverage action embeddings (Dulac-Arnold et al. 2015; He et al. 2015; Bajpai, Garg, and others 2018; Chandak et al. 2019; Tennenholtz and Mannor 2019). Building upon their ideas, we present a new objective for learning structure in the action space, and show that the performance of the policy resulting from using this inferred structure has bounded sub-optimality. Moreover, in contrast to their setup where the size of the action set is fixed, we consider the case of lifelong MDP, where the number of actions changes over time.

Mandel et al. (2017) and Boutilier et al. (2018) present the work most similar to ours. Mandel et al. (2017) consider the setting where humans can provide new actions to an RL system. The goal in their setup is to minimize human effort by querying for new actions only at states where new actions are most likely to boost performance. In comparison, our setup considers the case where the new actions become available through some external, unknown, process and the goal is to build learning algorithms that can efficiently adapt to such changes in the action set. Boutilier et al. (2018) laid the foundation for the stochastic action set MDP (SAS-MDP) setting where there is a fixed, finite, number of (base) actions and the available set of actions is a stochastically chosen subset of this base set. While SAS-MDPs can also be considered to have a 'changing action set', unlike their work there is no fixed maximum number for the available actions in our framework. Further, in their setup, there is a *possibility* that within a single long episode an agent can observe *all* possible actions it will ever encounter. In our set-up, this is never possible. As shown by Boutilier et al. (2018), SAS-MDPs can also be reduced to standard MDPs by extending the state space to include the set of available action. This cannot be done in our lifelong-MDP setup, as that would imply that the state-space is changing across episodes or the MDP is non-stationary. The works by Gabel and Riedmiller (2008) and Ferreira et al. (2017) also consider subsets of the base actions for DEC-MDPs and answer-set programming, respectively, but all the mentioned differences from the work by Boutilier et al. (2018) are also applicable here.

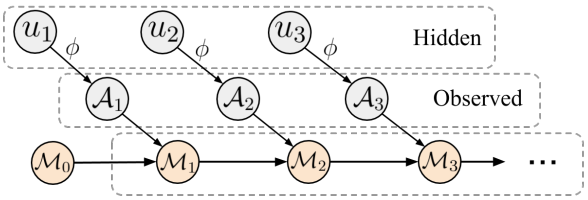

Figure 1: Illustration of a *lifelong MDP* where $\mathcal{M}_0$ is the base MDP. For every change k, $\mathcal{M}_K$ builds upon $\mathcal{M}_{k-1}$ by including the newly available set of actions $\mathcal{A}_k$. The internal structure in the space of actions is hidden and only a set of discrete actions is observed.

These differences let the proposed work better capture the challenges of lifelong learning, where the cardinality of the action set itself varies over time and an agent has to deal with actions that it has never dealt with before.

Lifelong Markov Decision Process

MDPs, the standard formalization of decision making problems, are not flexible enough to encompass lifelong learning problems wherein the action set size changes over time. In this section we extend the standard MDP framework to model this setting.

In real-world problems where the set of possible actions changes, there is often underlying structure in the set of all possible actions (those that are available, and those that may become available). For example, tutorial videos can be described by feature vectors that encode their topic, difficulty, length, and other attributes; in robot control tasks, primitive locomotion actions like left, right, up, and down could be encoded by their change to the Cartesian coordinates of the robot, etc. Critically, we will not assume that the agent knows this structure, merely that it exists. If actions are viewed from this perspective, then the set of all possible actions (those that are available at one point in time, and those that might become available at any time in the future) can be viewed as a vector-space, $\mathcal{E} \subseteq \mathbb{R}^d$.

To formalize the lifelong MDP, we first introduce the necessary variables that govern when and how new actions are added. We denote the episode number using τ. Let $I_\tau \in \{0, 1\}$ be a random variable that indicates whether a new set of actions are added or not at the start of episode τ, and let frequency $\mathcal{F} : \mathbb{N} \to [0, 1]$ be the associated probability distribution over episode count, such that $\Pr(I_\tau = 1) = \mathcal{F}(\tau)$. Let $U_\tau \in 2^{\mathcal{E}}$ be the random variable corresponding to the set of actions that is added before the start of episode τ. When $I_\tau = 1$, we assume that $U_\tau \neq \emptyset$, and when $I_\tau = 0$, we assume that $U_\tau = \emptyset$. Let $\mathcal{D}_\tau$ be the distribution of U_τ when $I_\tau = 1$, i.e., $U_\tau \sim \mathcal{D}_\tau$ if $I_\tau = 1$. We use $\mathcal{D}$ to denote the set $\{\mathcal{D}_\tau\}$ consisting of these distributions. Such a formulation using I_τ and $\mathcal{D}_\tau$ provides a fine control of when and how new actions can be incorporated. This allows modeling a large class of problems where both the distribution over the type of incorporated actions as well intervals between successive changes might be irregular. Often we will not require the exact episode number τ but instead require k, which denotes

the number of times the action set is changed.

Since we do not assume that the agent knows the structure associated with the action, we instead provide actions to the agent as a set of discrete entities, $\mathcal{A}_k$. To this end, we define ϕ to be a map relating the underlying structure of the new actions to the observed set of discrete actions $\mathcal{A}_k$ for all k, i.e., if the set of actions added is u_k, then $\mathcal{A}_k = \{\phi(e_i)|e_i \in u_k\}$. Naturally, for most problems of interest, neither the underlying structure $\mathcal{E}$, nor the set of distributions $\mathcal{D}$, nor the frequency of updates $\mathcal{F}$, nor the relation ϕ is known—the agent only has access to the observed set of discrete actions.

We now define the *lifelong Markov decision process* (L-MDP) as $\mathscr{L} = (\mathcal{M}_0, \mathcal{E}, \mathcal{D}, \mathcal{F})$, which extends a *base* MDP $\mathcal{M}_0 = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d_0)$. $\mathcal{S}$ is the set of all possible states that the agent can be in, called the state set. $\mathcal{A}$ is the discrete set of actions available to the agent, and for $\mathcal{M}_0$ we define this set to be empty, i.e., $\mathcal{A} = \emptyset$. When the set of available actions changes and the agent observes a new set of discrete actions, $\mathcal{A}_k$, then $\mathcal{M}_{k-1}$ transitions to $\mathcal{M}_k$, such that $\mathcal{A}$ in $\mathcal{M}_k$ is the set union of $\mathcal{A}$ in $\mathcal{M}_{k-1}$ and $\mathcal{A}_k$. Apart from the available actions, other aspects of the L-MDP remain the same throughout. An illustration of the framework is provided in Figure 1. We use $S_t \in \mathcal{S}$, $A_t \in \mathcal{A}$, and $R_t \in \mathbb{R}$ as random variables for denoting the state, action and reward at time $t \in \{0, 1, \dots\}$ within each episode. The first state, S_0, comes from an initial distribution, d_0, and the reward function $\mathcal{R}$ is defined to be only dependent on the state such that $\mathcal{R}(s) = \mathbf{E}[R_t|S_t = s]$ for all $s \in \mathcal{S}$. We assume that $R_t \in [-R_{\max}, R_{\max}]$ for some finite $R_{\max}$. The reward discounting parameter is given by $\gamma \in [0, 1)$. $\mathcal{P}$ is the state transition function, such that for all s, a, s', t, the function $\mathcal{P}(s, a, s')$ denotes the transition probability $P(s'|s, e)$, where $a = \phi(e)$.[1]

In the most general case, new actions could be completely arbitrary and have no relation to the ones seen before. In such cases, there is very little hope of lifelong learning by leveraging past experience. To make the problem more feasible, we resort to a notion of *smoothness* between actions. Formally, we assume that transition probabilities in an L-MDP are $\rho-$Lipschitz in the structure of actions, i.e., $\exists \rho > 0$ s.t.,

$$\forall s, s', e_i, e_j \;\; \|P(s'|s, e_i) - P(s'|s, e_j)\|_1 \leq \rho \|e_i - e_j\|_1. \tag{1}$$

For any given MDP $\mathcal{M}_k$ in $\mathscr{L}$, an agent's goal is to find a policy, π_k, that maximizes the expected sum of discounted future rewards. For any policy π_k, the corresponding state value function is $v^{\pi_k}(s) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_t|s, \pi_k]$.

Blessing of Changing Action Sets

Finding an optimal policy when the set of possible actions is large is difficult due to the curse of dimensionality. In the L-MDP setting this problem might appear to be exacerbated, as an agent must additionally adapt to the changing levels of possible performance as new actions become available. This raises the natural question: *as new actions become available, how much does the performance of an optimal policy change?* If it fluctuates significantly, can a lifelong learning agent succeed by continuously adapting its policy, or is it better to learn from scratch with every change to the action set?

To answer this question, consider an optimal policy, π_k^*, for MDP $\mathcal{M}_k$, i.e., an optimal policy when considering only policies that use actions that are available during the k^{th} episode. We now quantify how sub-optimal π_k^* is relative to the performance of a hypothetical policy, μ^*, that acts optimally given access to all possible actions.

Theorem 1. *In an L-MDP, let ϵ_k denote the maximum distance in the underlying structure of the closest pair of available actions, i.e., $\epsilon_k \coloneqq \sup_{a_i \in \mathcal{A}} \inf_{a_j \in \mathcal{A}} \|e_i - e_j\|_1$, then*

$$v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) \leq \frac{\gamma \rho \epsilon_k}{(1-\gamma)^2} R_{max}.$$

Proof. See Appendix B. □

With a bound on the maximum possible sub-optimality, Theorem 1 presents an important connection between achievable performances, the nature of underlying structure in the action space, and a property of available actions in any given $\mathcal{M}_k$. Using this, we can make the following conclusion.

Corollary 1. *Let $\mathcal{Y} \subseteq \mathcal{E}$ be the smallest closed set such that, $P(U_k \subseteq 2^{\mathcal{Y}}) = 1$. We refer to $\mathcal{Y}$ as the element-wise-support of U_k. If $\forall k$, the element-wise-support of U_k in an L-MDP is $\mathcal{E}$, then as $k \to \infty$ the sub-optimality vanishes. That is,*

$$\lim_{k \to \infty} v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) \to 0.$$

Proof. See Appendix B. □

Through Corollary 1, we can now establish that the change in optimal performance will eventually converge to zero as new actions are repeatedly added. An intuitive way to observe this result would be to notice that every new action that becomes available indirectly provides more information about the underlying, unknown, structure of $\mathcal{E}$. However, in the limit, as the size of the available action set increases, the information provided by each each new action vanishes and thus performance saturates.

Certainly, in practice, we can never have $k \to \infty$, but this result is still advantageous. Even when the underlying structure $\mathcal{E}$, the set of distributions $\mathcal{D}$, the change frequency $\mathcal{F}$, and the mapping relation ϕ are all *unknown*, it establishes the fact that the difference between the best performances in *successive changes* will remain bounded and will not fluctuate arbitrarily. This opens up new possibilities for developing algorithms that do not need to start from scratch after new actions are added, but rather can build upon their past experiences using updates to their existing policies that efficiently leverage estimates of the structure of $\mathcal{E}$ to adapt to new actions.

Learning with Changing Action Sets

Theorem 1 characterizes what *can be* achieved in principle, however, it does not specify *how* to achieve it—how to find π_k^* efficiently. Using any parameterized policy, π, which acts

[1] For notational ease, (a) we overload symbol P for representing both probability mass and density; (b) we assume that the state set is finite, however, our primary results extend to MDPs with continuous states.

directly in the space of observed actions, suffers from one key practical drawback in the L-MDP setting. That is, the parameterization is deeply coupled with the number of actions that are available. That is, not only is the meaning of each parameter coupled with the number of actions, but often the number of parameters that the policy has is dependent on the number of possible actions. This makes it unclear how the policy should be adapted when additional actions become available. A trivial solution would be to ignore the newly available actions and continue only using the previously available actions. However, this is clearly myopic, and will prevent the agent from achieving the better long term returns that might be possible using the new actions.

To address this parameterization-problem, instead of having the policy, π, act directly in the observed action space, $\mathcal{A}$, we propose an approach wherein the agent reasons about the underlying structure of the problem in a way that makes its policy parameterization invariant to the number of actions that are available. To do so, we split the policy parameterization into two components. The first component corresponds to the state conditional policy responsible for making the decisions, $\beta : \mathcal{S} \times \hat{\mathcal{E}} \to [0, 1]$, where $\hat{\mathcal{E}} \in \mathbb{R}^d$. The second component corresponds to $\hat{\phi} : \hat{\mathcal{E}} \times \mathcal{A} \to [0, 1]$, an estimator of the relation ϕ, which is used to map the output of β to an action in the set of available actions. That is, an $E_t \in \hat{\mathcal{E}}$ is sampled from $\beta(S_t, \cdot)$ and then $\hat{\phi}(E_t)$ is used to obtain the action A_t. Together, β and $\hat{\phi}$ form a complete policy, and $\hat{\mathcal{E}}$ corresponds to the inferred structure in action space.

One of the prime benefits of estimating ϕ with $\hat{\phi}$ is that it makes the parameterization of β invariant to the cardinality of the action set—changing the number of available actions does not require changing the number of parameters of β. Instead, only the parameterization of $\hat{\phi}$, the estimator of the underlying structure in action space, must be modified when new actions become available. We show next that the update to the parameters of $\hat{\phi}$ can be performed using *supervised learning* methods that are independent of the reward signal and thus typically more efficient than RL methods.

While our proposed parameterization of the policy using both β and $\hat{\phi}$ has the advantages described above, the performance of β is now constrained by the quality of $\hat{\phi}$, as in the end $\hat{\phi}$ is responsible for selecting an action from $\mathcal{A}$. Ideally we want $\hat{\phi}$ to be such that it lets β be both: (a) invariant to the cardinality of the action set for practical reasons and (b) as expressive as a policy, π, explicitly parameterized for the currently available actions. Similar trade-offs have been considered in the context of learning optimal state-embeddings for representing sub-goals in hierarchical RL (Nachum et al. 2018). For our lifelong learning setting, we build upon their method to efficiently estimate $\hat{\phi}$ in a way that provides bounded sub-optimality. Specifically, we make use of an additional *inverse dynamics* function, φ, that takes as input two states, s and s', and produces as output a prediction of which $e \in \mathcal{E}$ caused the transition from s to s'. Since the agent does not know ϕ, when it observes a transition from s to s' via action a, it does *not* know which e caused this transition. So, we cannot train φ to make good predictions using the actual action, e, that caused the transition. Instead, we use $\hat{\phi}$ to transform the prediction of φ from $e \in \mathcal{E}$ to $a \in \mathcal{A}$, and train both φ and $\hat{\phi}$ so that this process accurately predicts which action, a, caused the transition from s to s'. Moreover, rather than viewing φ as a deterministic function mapping states s and s' to predictions e, we define φ to be a *distribution* over $\mathcal{E}$ given two states, s and s'.

For any given $\mathcal{M}_k$ in L-MDP $\mathscr{L}$, let β_k and $\hat{\phi}_k$ denote the two components of the overall policy and let π_k^{**} denote the best overall policy that can be represented using some fixed $\hat{\phi}_k$. The following theorem bounds the sub-optimality of π_k^{**}.

Theorem 2. *For an L-MDP $\mathcal{M}_k$, If there exists a $\varphi : S \times S \times \hat{\mathcal{E}} \to [0, 1]$ and $\hat{\phi}_k : \hat{\mathcal{E}} \times \mathcal{A} \to [0, 1]$ such that*

$$\sup_{s \in \mathcal{S}, a \in \mathcal{A}} KL\Big(P(S_{t+1}|S_t = s, A_t = a) \| P(S_{t+1}|S_t = s, A_t = \hat{A})\Big) \leq \delta_k^2/2, \quad (2)$$

where $\hat{A} \sim \hat{\phi}_k(\cdot|\hat{E})$ and $\hat{E} \sim \varphi(\cdot|S_t, S_{t+1})$, then

$$v^{\mu^*}(s_0) - v^{\pi_k^{**}}(s_0) \leq \frac{\gamma\left(\rho\epsilon_k + \delta_k\right)}{(1-\gamma)^2} R_{max}.$$

Proof. See Appendix B. □

By quantifying the impact $\hat{\phi}$ has on the sub-optimality of achievable performance, Theorem 2 provides the necessary constraints for estimating $\hat{\phi}$. At a high level, Equation (2) ensures $\hat{\phi}$ to be such that it can be used to generate an action corresponding to any s to s' transition. This allows β to leverage $\hat{\phi}$ and choose the required action that induces the state transition needed for maximizing performance. Thereby, following (2), sub-optimality would be minimized if $\hat{\phi}$ and φ are optimized to reduce the supremum of KL divergence over all s and a. In practice, however, the agent does not have access to all possible states, rather it has access to a limited set of samples collected from interactions with the environment. Therefore, instead of the supremum, we propose minimizing the average over all s and a from a set of observed transitions,

$$\mathcal{L}(\hat{\phi}, \varphi) := \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} P(s, a) KL\left(P(s'|s, a) \| P(s'|s, \hat{a})\right). \quad (3)$$

Equation (3) suggests that $\mathcal{L}(\hat{\phi}, \varphi)$ would be minimized when $\hat{a}$ equals a, but using (3) directly in the current form is inefficient as it requires computing KL over all probable $s' \in \mathcal{S}$ for a given s and a. To make it practical, we make use of the following property.

Property 1. *For some constant C, $-\mathcal{L}(\hat{\phi}, \varphi)$ is lower bounded by*

$$\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s, a, s') \Bigg(\mathbf{E}\left[\log \hat{\phi}(\hat{a}|\hat{e}) \Big| \hat{e} \sim \varphi(\cdot|s, s')\right] - KL\Big(\varphi(\hat{e}|s, s') \Big\| P(\hat{e}|s, s')\Big)\Bigg) + C. \quad (4)$$

Proof. See Appendix C. □

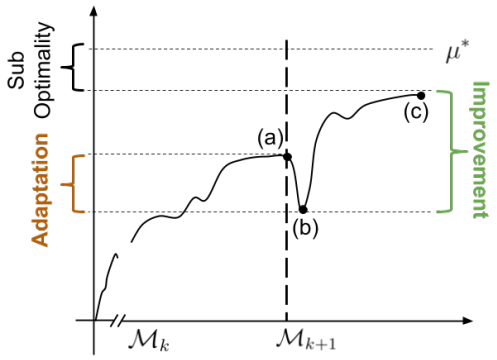

Figure 2: An illustration of a typical performance curve for a lifelong learning agent. The point (a) corresponds to the performance of the current policy in $\mathcal{M}_k$. The point (b) corresponds to the performance drop resulting as a consequence of adding new actions. We call the phase between (a) and (b) as the adaptation phase, which aims at minimizing this drop when adapting to new set of actions. The point (c) corresponds to the improved performance in $\mathcal{M}_{k+1}$ by optimizing the policy to leverage the new set of available actions. μ^* represents the best performance of the hypothetical policy which has access to the entire structure in the action space.

As minimizing $\mathcal{L}(\hat{\phi}, \varphi)$ is equivalent to maximizing $-\mathcal{L}(\hat{\phi}, \varphi)$, we consider maximizing the lower bound obtained from Property 1. In this form, it is now practical to optimize (4) just by using the observed (s, a, s') samples. As this form is similar to the objective for variational auto-encoder, inner expectation can be efficiently optimized using the reparameterization trick (Kingma and Welling 2013). $P(\hat{e}|s, s')$ is the prior on $\hat{e}$, and we treat it as a hyper-parameter that allows the KL to be computed in closed form.

Importantly, note that this optimization procedure only requires individual transitions, s, a, s', and is independent of the reward signal. Hence, at its core, it is a supervised learning procedure. This means that learning good parameters for $\hat{\phi}$ tends to require far fewer samples than optimizing β (which is an RL problem). This is beneficial for our approach because $\hat{\phi}$, the component of the policy where new parameters need to be added when new actions become available, can be updated efficiently. As both β and φ are invariant to action cardinality, they do not require new parameters when new actions become available. Additional implementation level details are available in Appendix F.

Algorithm

When a new set of actions, $\mathcal{A}_{k+1}$, becomes available, the agent should leverage the existing knowledge and quickly adapt to the new action set. Therefore, during every change in $\mathcal{M}_k$, the ongoing best components of the policy, β^*_{k-1} and ϕ^*_{k-1}, in $\mathcal{M}_{k-1}$ are carried over, i.e., $\beta_k := \beta^*_{k-1}$ and $\hat{\phi}_k := \hat{\phi}^*_{k-1}$. For lifelong learning, the following property illustrates a way to organize the learning procedure so as to minimize the sub-optimality in each $\mathcal{M}_k$, for all k.

Property 2. *(Lifelong Adaptation and Improvement) In an L-MDP, let Δ denote the difference of performance between v^{μ^*} and the best achievable using our policy parameterization, then the overall sub-optimality can be expressed as,*

$$v^{\mu^*}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) = \underbrace{\sum_{k=1}^{\infty}\left(v^{\beta_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) - v^{\beta_k\hat{\phi}_k}_{\mathcal{M}_k}(s_0)\right)}_{\textit{Adaptation}} + \underbrace{\sum_{k=1}^{\infty}\left(v^{\beta^*_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) - v^{\beta_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0)\right)}_{\textit{Policy Improvement}} + \Delta,$$

where $\mathcal{M}_k$ is used in the subscript to emphasize the respective MDP in $\mathscr{L}$. ***Proof:*** *See Appendix D.*

Property 2 illustrates a way to understand the impact of β and $\hat{\phi}$ by splitting the learning process into an adaptation phase and a policy improvement phase. These two iterative phases are the crux of our algorithm for solving an L-MDP $\mathscr{L}$. Based on this principle, we call our algorithm LAICA: *lifelong adaptation and improvement for changing actions*. Due to space constraints, we now briefly discuss the LAICA algorithm; a detailed description with pseudocode is presented in Appendix E.

Whenever new actions become available, adaptation is prone to cause a performance drop as the agent has no information about when to use the new actions, and so its initial uses of the new actions may be at inappropriate times. Following Property 1, we update $\hat{\phi}$ so as to efficiently infer the underlying structure and minimize this drop. That is, for every $\mathcal{M}_k$, $\hat{\phi}_k$ is first adapted to $\hat{\phi}^*_k$ in the adaptation phase by adding more parameters for the new set of actions and then optimizing (4). After that, $\hat{\phi}^*_k$ is fixed and β_k is improved towards β^*_k in the policy improvement phase, by updating the parameters of β_k using the policy gradient theorem (Sutton et al. 2000). These two procedures are performed sequentially whenever $\mathcal{M}_{k-1}$ transitions to $\mathcal{M}_k$, for all k, in an L-MDP $\mathscr{L}$. An illustration of the procedure is presented in Figure 2.

A step-by-step pseudo-code for the LAICA algorithm is available in Algorithm 1, Appendix E. The crux of the algorithm is based on the iterative adapt and improve procedure obtained from Property 2.

Empirical Analysis

In this section, we aim to empirically compare the following methods,

- Baseline(1): The policy is re-initialised and the agent learns from scratch after every change.
- Baseline(2): New parameters corresponding to new actions are added/stacked to the existing policy (and previously learned parameters are carried forward as-is).
- LAICA(1): The proposed approach that leverages the structure in the action space. To act in continuous space of inferred structure, we use DPG (Silver et al. 2014) to optimize β.

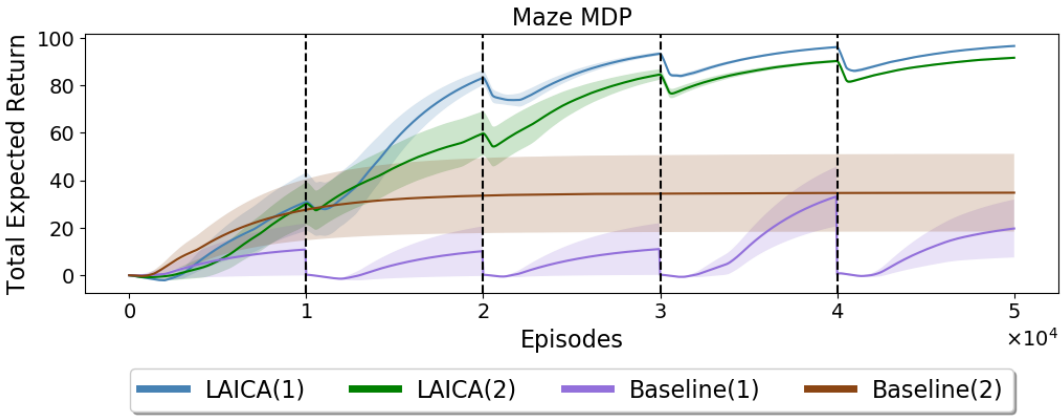

Figure 3: Lifelong learning experiments with a changing set of actions in the maze domain. The learning curves correspond to the running mean of the best performing setting for each of the algorithms. The shaded regions correspond to standard error obtained using 10 trials. Vertical dotted bars indicate when the set of actions was changed.

- LAICA(2): A variant of LAICA which uses an actor-critic (Sutton and Barto 2018) to optimize β.

To demonstrate the effectiveness of our proposed method(s) on lifelong learning problems, we consider a maze environment and two domains corresponding to real-world applications, all with a large set of changing actions. For each of these domains, the total number of actions were randomly split into five equal sets. Initially, the agent only had the actions available in the first set and after every change the next set of actions was made available additionally. In the following paragraphs we briefly outline the domains; full details are deferred to Appendix F.

Maze Domain. As a proof-of-concept, we constructed a continuous-state maze environment where the state is comprised of the coordinates of the agent's location and its objective is to reach a fixed goal state. The agent has a total of 256 actions corresponding to displacements in different directions of different magnitudes. This domain provides a simple yet challenging testbed that requires solving a long horizon task using a large, changing action set, in presence of a single goal reward.

Case Study: Real-World Recommender Systems. We consider the following two real-world applications of large-scale recommender systems that require decision making over multiple time steps and where the number of possible decisions varies over the lifetime of the system.

- A web-based video-tutorial platform, that has a recommendation engine to suggest a series of tutorial videos. The aim is to meaningfully engage the users in a learning activity. In total, 1498 tutorials were considered for recommendation.
- A professional multi-media editing software, where sequences of tools inside the software need to be recommended. The aim is to increase user productivity and assist users in quickly achieving their end goal. In total, 1843 tools were considered for recommendation.

For both of these applications, an existing log of user's click stream data was used to create an n-gram based MDP model for user behavior (Shani, Heckerman, and Brafman 2005). Sequences of user interaction were aggregated to obtain over 29 million clicks and 1.75 billion user clicks for the tutorial recommendation and the tool recommendation task, respectively. The MDP had continuous state-space, where each state consisted of the feature descriptors associated with each item (tutorial or tool) in the current n-gram.

Results.

The plots in Figures 3 and 4 present the evaluations on the domains considered. The advantage of LAICA over Baseline(1) can be attributed to its policy parameterization. The decision making component of the policy, β, being invariant to the action cardinality can be readily leveraged after every change without having to be re-initialized. This demonstrates that efficiently re-using past knowledge can improve data efficiency over the approach that learns from scratch every time.

Compared to Baseline(2), which also does not start from scratch and reuses existing policy, we notice that the variants of LAICA algorithm still perform favorably. As evident from the plots in Figures 3 and 4, while Baseline(2) does a good job of preserving the existing policy, it fails to efficiently capture the benefit of new actions. While the policy parameters in both LAICA and Baseline(2) are improved using policy gradients, the superior performance of LAICA can be attributed to the adaptation procedure incorporated in LAICA which aims at efficiently inferring the underlying structure in the space of actions. Overall LAICA(2) performs almost twice as well as both the baselines on all of the tasks considered. In the maze domain, even the best setting for Baseline(2) performed inconsistently. Due to the sparse reward nature of the task, which only had a big positive reward on reaching goal, even the best setting for Baseline(2) failed on certain

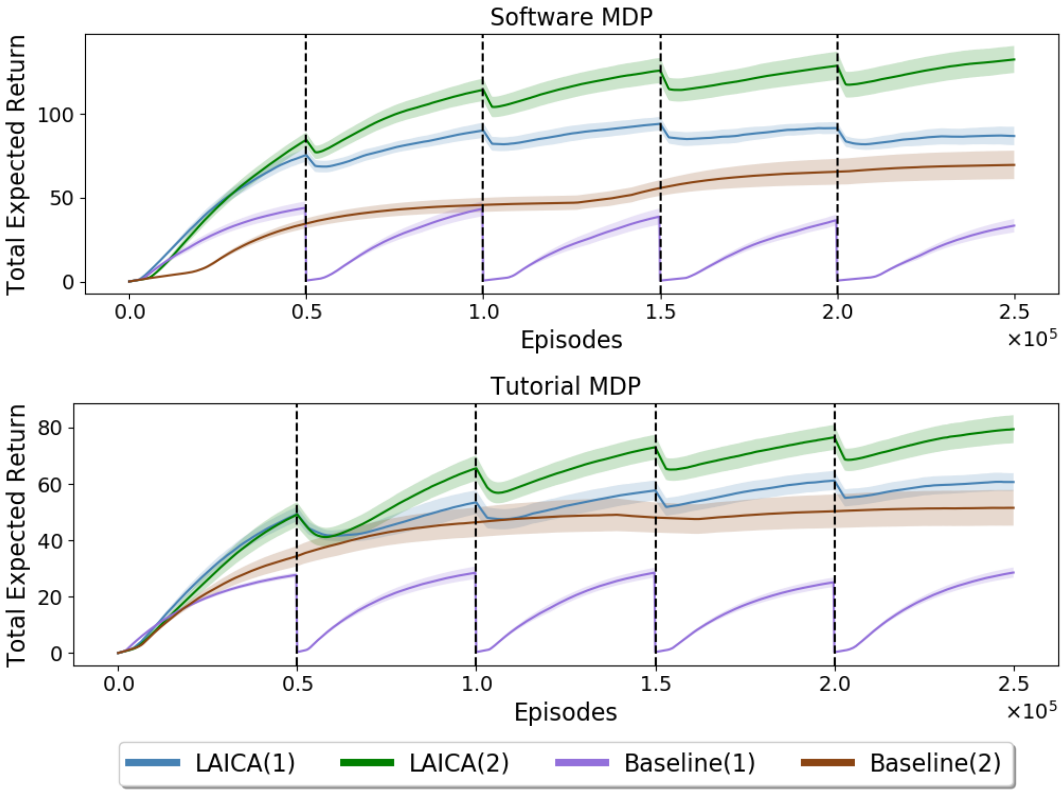

Figure 4: Lifelong learning experiments with a changing set of actions in the recommender system domains. The learning curves correspond to the running mean of the best performing setting for each of the algorithms. The shaded regions correspond to standard error obtained using 10 trials. Vertical dotted bars indicate when the set of actions was changed.

trials, resulting in high variance.

Note that even before the first addition of the new set of actions, the proposed method performs better than the baselines. This can be attributed to the fact that the proposed method efficiently leverages the underlying structure in the action set and thus learns faster. Similar observations have been made previously (Dulac-Arnold et al. 2015; He et al. 2015; Bajpai, Garg, and others 2018).

In terms of computational cost, the proposed method updates the inverse dynamics model and the underlying action structure only when there is a change in the action set (Algorithm 1). Therefore, compared to the baselines, no extra computational cost is required for training at *each* time-step. However, the added computational cost does impact the *overall* learning process and is proportional to the number of times new actions are introduced.

Discussion and Conclusion

In this work we established first steps towards developing the lifelong MDP setup for dealing with action sets that change over time. Our proposed approach then leveraged the structure in the action space such that an existing policy can be efficiently adapted to the new set of available actions. Superior performances on both synthetic and large-scale real-world environments demonstrate the benefits of the proposed LAICA algorithm.

To the best of our knowledge, this is the first work to take a step towards addressing the problem of lifelong learning with a changing action set. We hope that this brings more attention to such understudied problems in lifelong learning. There are several important challenges open for future investigation.

In many real-world applications, often due to external factors, some actions are removed over time as well. For example, if a medicine becomes outdated, if a product is banned, etc. While our applications were devoid of this aspect, the proposed algorithm makes use of a policy parameterization that is invariant to the cardinality of the action set, and thus can support both addition and removal. Our proposed policy decomposition method can still be useful for selecting an available action whose impact on the state transition is most similar to the removed action.

There can be several applications where new actions that are added over time have no relation to the previously observed actions. For example, completely new product categories, tutorial videos on new topics, etc. In such cases, it is unclear how to leverage past information efficiently. We do not expect our proposed method to work well in such settings.

Acknowledgement

The research was supported by and partially conducted at Adobe Research. We are also immensely grateful to the three anonymous reviewers who shared their insights and feedback.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 22–31. JMLR. org.

Bajpai, A. N.; Garg, S.; et al. 2018. Transfer of deep reactive policies for mdp planning. In *Advances in Neural Information Processing Systems*, 10965–10975.

Boutilier, C.; Cohen, A.; Daniely, A.; Hassidim, A.; Mansour, Y.; Meshi, O.; Mladenov, M.; and Schuurmans, D. 2018. Planning and learning with stochastic action sets. In *IJCAI*.

Chandak, Y.; Theocharous, G.; Kostas, J.; Jordan, S.; and Thomas, P. S. 2019. Learning action representations for reinforcement learning. *International Conference on Machine Learning*.

Chen, Z., and Liu, B. 2016. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10(3):1–145.

Devroye, L.; Györfi, L.; Lugosi, G.; and Walk, H. 2017. On the measure of voronoi cells. *Journal of Applied Probability*.

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

Dulac-Arnold, G.; Evans, R.; van Hasselt, H.; Sunehag, P.; Lillicrap, T.; Hunt, J.; Mann, T.; Weber, T.; Degris, T.; and Coppin, B. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

Ferreira, L. A.; Bianchi, R. A.; Santos, P. E.; and de Mantaras, R. L. 2017. Answer set programming for non-stationary markov decision processes. *Applied Intelligence* 47(4):993–1007.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* 3(4):128–135.

Gabel, T., and Riedmiller, M. 2008. Reinforcement learning for dec-mdps with changing action sets and partially ordered dependencies. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, 1333–1336. International Foundation for Autonomous Agents and Multiagent Systems.

Gajane, P.; Ortner, R.; and Auer, P. 2018. A sliding-window algorithm for markov decision processes with arbitrarily changing rewards and transitions. *arXiv preprint arXiv:1805.10066*.

Gupta, A.; Mendonca, R.; Liu, Y.; Abbeel, P.; and Levine, S. 2018. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, 5302–5311.

He, J.; Chen, J.; He, X.; Gao, J.; Li, L.; Deng, L.; and Ostendorf, M. 2015. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3.

Kakade, S., and Langford, J. 2002. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, 267–274.

Kearns, M., and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine learning* 49(2-3):209–232.

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

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114(13):3521–3526.

Konidaris, G.; Osentoski, S.; and Thomas, P. S. 2011. Value function approximation in reinforcement learning using the fourier basis. In *AAAI*, volume 6, 7.

Lopez-Paz, D., et al. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 6467–6476.

Mandel, T.; Liu, Y.-E.; Brunskill, E.; and Popović, Z. 2017. Where to add actions in human-in-the-loop reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2018. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*.

Neu, G. 2013. Online learning in non-stationary markov decision processes. *CoRR*.

Pirotta, M.; Restelli, M.; Pecorino, A.; and Calandriello, D. 2013. Safe policy iteration. In *International Conference on Machine Learning*, 307–315.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Ruvolo, P., and Eaton, E. 2013. Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, 507–515.

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An mdp-based recommender system. *Journal of Machine Learning Research* 6(Sep):1265–1295.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *ICML*.

Silver, D. L.; Yang, Q.; and Li, L. 2013. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*.

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

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1-2):181–211.

Tennenholtz, G., and Mannor, S. 2019. The natural language of actions. *International Conference on Machine Learning*.

Thrun, S. 1998. Lifelong learning algorithms. In *Learning to learn*. Springer. 181–209.

Wang, J.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2017. Learning to reinforcement learn. arxiv 1611.05763.

Xu, Z.; van Hasselt, H. P.; and Silver, D. 2018. Meta-gradient reinforcement learning. In *Advances in neural information processing systems*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 3987–3995. JMLR. org.

Lifelong Learning with a Changing Action Set (Supplementary Material)

A: Preliminary

For the purpose of our results, we would require bounding the shift in the state distribution between two policies. Techniques for doing so has been previously studied in literature (Kakade and Langford 2002; Kearns and Singh 2002; Pirotta et al. 2013; Achiam et al. 2017). Specifically, we cover this preliminary result based on the work by Achiam et al. (2017).

The discounted state distribution, for all $s \in \mathcal{S}$, for a policy π is given by,

$$d_\pi(s) = (1-\gamma)\sum_t \gamma^t P(S_t = s|\pi). \tag{5}$$

Let the shift in state distribution between any given policies π_1 and π_2 be denoted as $D(\pi_1, \pi_2)$, such that

$$\begin{aligned} D(\pi_1,\pi_2) &= \int_{\mathcal{S}} |d_{\pi_1}(s) - d_{\pi_2}(s)|\,\mathrm{d}s \\ &= \int_{\mathcal{S}} \left|(1-\gamma)\sum_t \gamma^t P(S_t = s|\pi_1) - (1-\gamma)\sum_t \gamma^t P(S_t = s|\pi_2)\right| \mathrm{d}s. \end{aligned} \tag{6}$$

For any policy π, let P^π denote the matrix corresponding to transition probabilities as a result of π. Then (6) can be re-written as,

$$\begin{aligned} D(\pi_1,\pi_2) &= \left\|(1-\gamma)(1-\gamma P^{\pi_1})^{-1}d_0 - (1-\gamma)(1-\gamma P^{\pi_2})^{-1}d_0\right\|_1 \\ &= \left\|(1-\gamma)\big((1-\gamma P^{\pi_1})^{-1} - (1-\gamma P^{\pi_2})^{-1}\big)d_0\right\|_1. \end{aligned} \tag{7}$$

To simplify (7), let $G_1 = (1-\gamma P^{\pi_1})^{-1}$ and $G_2 = (1-\gamma P^{\pi_2})^{-1}$.
Then, $G_1 - G_2 = G_1(G_2^{-1} - G_1^{-1})G_2$, and therefore (7) can be written as,

$$\begin{aligned} D(\pi_1,\pi_2) &= \left\|(1-\gamma)\big((1-\gamma P^{\pi_1})^{-1}((1-\gamma P^{\pi_2}) - (1-\gamma P^{\pi_1}))(1-\gamma P^{\pi_2})^{-1}\big)d_0\right\|_1 \\ &= \left\|(1-\gamma)\big((1-\gamma P^{\pi_1})^{-1}(\gamma P^{\pi_1} - \gamma P^{\pi_2})(1-\gamma P^{\pi_2})^{-1}\big)d_0\right\|_1 \\ &= \left\|\big((1-\gamma P^{\pi_1})^{-1}(\gamma P^{\pi_1} - \gamma P^{\pi_2})\big)d_{\pi_2}\right\|_1. \end{aligned} \tag{8}$$

Note that using matrix L1 norm,

$$\left\|(1-\gamma P^{\pi_1})^{-1}\right\|_1 = \left\|\sum_t (\gamma P^{\pi_1})^t\right\|_1 \le \sum_t \gamma^t \left\|P^{\pi_1^t}\right\|_1 = \sum_t \gamma^t \cdot \mathbf{1} = (1-\gamma)^{-1}. \tag{9}$$

Combining (9) and (8),

$$D(\pi_1,\pi_2) \le \gamma(1-\gamma)^{-1}\left\|(P^{\pi_1} - P^{\pi_2})d_{\pi_2}\right\|_1.$$

B: Sub-Optimality

Theorem 1. *In an L-MDP, let ϵ_k denote the maximum distance in the underlying structure of the closest pair of available actions, i.e., $\epsilon_k \coloneqq \sup_{a_i \in \mathcal{A}} \inf_{a_j \in \mathcal{A}} \|e_i - e_j\|_1$, then*

$$v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) \le \frac{\gamma\rho\epsilon_k}{(1-\gamma)^2} R_{max}.$$

Proof. We begin by defining μ_k^* to be a policy where the actions of the policy μ^* is restricted to the actions available in $\mathcal{M}_k$. That is, any action e_i from μ^* is mapped to the closest e_j, where $a = \phi(e_j)$ is in the available action set. Notice that the best policy, π_k^*, using the available set of actions is always better than or equal to μ_k^*, i.e., $v^{\mu_k^*} \le v^{\pi_k^*}$. Therefore ,

$$v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) \le \left|v^{\mu^*}(s_0) - v^{\mu_k^*}(s_0)\right|. \tag{10}$$

On expanding the $v(s_0)$ corresponding for both the policies in (10) using (5),

$$\begin{aligned} v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0)| &\le \left|(1-\gamma)^{-1}\int_{\mathcal{S}} d_{\mu^*}(s)R(s)\mathrm{d}s - (1-\gamma)^{-1}\int_{\mathcal{S}} d_{\mu_k^*}(s)R(s)\mathrm{d}s\right| \\ &= \left|(1-\gamma)^{-1}\int_{\mathcal{S}} \Big(d_{\mu^*}(s) - d_{\mu_k^*}(s)\Big)R(s)\mathrm{d}s\right|. \end{aligned} \tag{11}$$

We can then upper bound (11) by taking the maximum possible reward common,

$$\begin{aligned} v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) &\leq (1-\gamma)^{-1} \int_{\mathcal{S}} \left| \left(d_{\mu^*}(s) - d_{\mu_k^*}(s) \right) \right| |R(s)| \, \mathrm{d}s \\ &\leq (1-\gamma)^{-1} R_{\max} \int_{\mathcal{S}} \left| d_{\mu^*}(s) - d_{\mu_k^*}(s) \right| \mathrm{d}s \\ &= (1-\gamma)^{-1} R_{\max} D(\mu^*, \mu_k^*) \\ &\leq \gamma (1-\gamma)^{-2} R_{\max} \left\| (P^{\mu_k} - P^{\mu_k^*}) d_{\mu_k^*} \right\|_1 \end{aligned} \tag{12}$$

For any action $\bar{e}$ taken by the policy μ^*, let $\bar{e}_k$ denote the action for μ_k^* obtained by mapping $\bar{e}$ to the closest action in the available set, then expanding (12), we get,

$$\begin{aligned} v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) &\leq \gamma (1-\gamma)^{-2} R_{\max} \left(\sup_s \left\| P^{\mu_k}(s) - P^{\mu_k^*}(s) \right\|_1 \right) \\ &\leq \gamma (1-\gamma)^{-2} R_{\max} \left(\sup_{s,s'} \left| \int_{\bar{e}} (P(s'|s,\bar{e}) - P(s'|s,\bar{e}_k)) \mu^*(\bar{e}|s) \mathrm{d}\bar{e} \right| \right) \\ &\leq \gamma (1-\gamma)^{-2} R_{\max} \left(\sup_{s,s',\bar{e}} \left| P(s'|s,\bar{e}) - P(s'|s,\bar{e}_k) \right| \right). \end{aligned} \tag{13}$$

From the Lipschitz condition (1), we know that $|P(s'|s,\bar{e}) - P(s'|s,\bar{e}_k)| \leq \rho \|\bar{e} - \bar{e}_k\|_1$. As $\bar{e}_k$ corresponds to the closest available action for $\bar{e}$, the maximum distance for $|\bar{e} - \bar{e}_k\|_1$ is bounded by ϵ_k. Combining (13) with these two observations, we get the desired result,

$$v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) \leq \frac{\gamma \rho \epsilon_k}{(1-\gamma)^2} R_{\max}.$$

□

Corollary 1. *Let $\mathcal{Y} \subseteq \mathcal{E}$ be the smallest closed set such that, $P(U_k \subseteq 2^{\mathcal{Y}}) = 1$. We refer to $\mathcal{Y}$ as the element-wise-support of U_k. If $\forall k$, the element-wise-support of U_k in an L-MDP is $\mathcal{E}$, then as $k \to \infty$ the sub-optimality vanishes. That is,*

$$\lim_{k \to \infty} v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) \to 0.$$

Proof. Let $X_1, ..., X_n$ be independent identically distributed random vectors in $\mathcal{E}$. Let X_i define a partition of $\mathcal{E}$ in n sets $V_1, ...V_n$, such that V_i contains all points in $\mathcal{E}$ whose nearest neighbor among $X_1, ..., X_n$ is X_i. Each such V_i forms a *Voronoi cell*. Now using the condition on full element-wise support, we know from the distribution free result by Devroye et al. (2017) that the diameter(V_i) converges to 0 at the rate $n^{-1/d}$ as $n \to \infty$ (Theorem 4, Devroye et al. (2017)). As ϵ_k corresponds to the maximum distance between closest pair of points in $\mathcal{E}$, $\epsilon_k \leq \sup_i \text{diameter}(V_i)$. Therefore, when $k \to \infty$ then $n \to \infty$; consequently $\epsilon_k \to 0$ and thus $v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) \to 0$.

□

Theorem 2. *For an L-MDP $\mathcal{M}_k$, If there exists a $\varphi : S \times S \times \hat{\mathcal{E}} \to [0,1]$ and $\hat{\phi}_k : \hat{\mathcal{E}} \times \mathcal{A} \to [0,1]$ such that*

$$\sup_{s \in \mathcal{S}, a \in \mathcal{A}} KL \left(P(S_{t+1}|S_t = s, A_t = a) \| P(S_{t+1}|S_t = s, A_t = \hat{A}) \right) \leq \delta_k^2/2, \tag{14}$$

where $\hat{A} \sim \hat{\phi}_k(\cdot|\hat{E})$ and $\hat{E} \sim \varphi(\cdot|S_t, S_{t+1})$, then

$$v^{\mu^*}(s_0) - v^{\pi_k^{**}}(s_0) \leq \frac{\gamma \left(\rho \epsilon_k + \delta_k \right)}{(1-\gamma)^2} R_{max}.$$

Proof. We begin by noting that,

$$v^{\mu^*}(s_0) - v^{\pi_k^{**}}(s_0) = v^{\mu^*}(s_0) - v^{\pi_k^*}(s_0) + v^{\pi_k^*}(s_0) - v^{\pi_k^{**}}(s_0).$$

Using Theorem (1),

$$v^{\mu^*}(s_0) - v^{\pi_k^{**}}(s_0) \leq \frac{\gamma \rho \epsilon_k}{(1-\gamma)^2} R_{\max} + \left(v^{\pi_k^*}(s_0) - v^{\pi_k^{**}}(s_0) \right). \tag{15}$$

Now we focus on bounding the last two terms in (15). Following steps similar to (11) and (12) it can bounded as,

$$\begin{aligned}
v^{\pi_k^*}(s_0) - v^{\pi_k^{**}}(s_0) &\leq \left| (1-\gamma)^{-1} R_{\max} \int_{\mathcal{S}} \left(d_{\pi_k^*}(s) - d_{\pi_k^{**}}(s) \right) \mathrm{d}s \right| \\
&\leq (1-\gamma)^{-1} R_{\max} \int_{\mathcal{S}} \left| d_{\pi_k^*}(s) - d_{\pi_k^{**}}(s) \right| \mathrm{d}s \\
&= (1-\gamma)^{-1} R_{\max} D(\pi_k^*, \pi_k^{**}) \\
&= \gamma (1-\gamma)^{-2} R_{\max} \left\| (P^{\pi_k^*} - P^{\pi_k^{**}}) d_{\pi_k^{**}} \right\|_1 \\
&\leq \gamma (1-\gamma)^{-2} R_{\max} \left(\mathbb{E}\left[2TV\left(P^{\pi_k^*}(s'|s) \| P^{\pi_k^{**}}(s'|s) \right) \middle| s \sim d_{\pi_k^{**}} \right] \right),
\end{aligned}$$

where TV stands for total variation distance. Using Pinsker's inequality,

$$\begin{aligned}
v^{\pi_k^*}(s_0) - v^{\pi_k^{**}}(s_0) &\leq \gamma (1-\gamma)^{-2} R_{\max} \left(\sup_s \sqrt{2KL\left(P^{\pi_k^*}(s'|s) \| P^{\pi_k^{**}}(s'|s) \right)} \right), \\
&\leq \gamma (1-\gamma)^{-2} R_{\max} \left(\sup_{s,a} \sqrt{2KL\left(P(s'|s,a) \| P(s'|s,\hat{a}) \right)} \right),
\end{aligned}$$

where, $a \sim \pi_k^*$ and $\hat{a} \sim \pi_k^{**}$. As condition (14) ensures that maximum KL divergence error between an actual a and an action that can be induced through $\hat{\phi}_k$ for transitioning from s to s' is bounded by $\delta_k^2/2$, we get the desired result,

$$v^{\pi_k^*}(s_0) - v^{\pi_k^{**}}(s_0) \leq \frac{\gamma \delta_k}{(1-\gamma)^2} R_{\max}. \tag{16}$$

Therefore taking the union bound on (15) and (16), we get the desired result

$$v^{\mu^*}(s_0) - v^{\pi_k^{**}}(s_0) \leq \frac{\gamma \left(\rho \epsilon_k + \delta_k \right)}{(1-\gamma)^2} R_{\max}.$$

□

C: Lower Bound Objective For Adaptation

$$\begin{aligned}
\mathcal{L}(\hat{\phi}, \varphi) &= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} P(s,a) \mathrm{KL}\left(P(s'|s,a) \| P(s'|s,\hat{a}) \right) \\
&= -\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} P(s,a) \sum_{s' \in \mathcal{S}} P(s'|s,a) \log P(s'|s,\hat{a}) + C_1
\end{aligned}$$

where C_1 is a constant corresponding to the entropy term in KL that is independent of $\hat{a}$. Continuing, we take the negative on both sides,

$$\begin{aligned}
-\mathcal{L}(\hat{\phi}, \varphi) &= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s,a,s') \log P(s'|s,\hat{a}) - C_1 \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s,a,s') \log \frac{P(s,\hat{a},s')}{P(s,\hat{a})} - C_1 \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s,a,s') \log \frac{P(s,\hat{a},s')}{\sum_{s' \in \mathcal{S}} P(s,\hat{a},s')} - C_1 \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s,a,s') \left[\log P(s,\hat{a},s') - \log Z \right] - C_1
\end{aligned}$$

where $Z = \sum_{s' \in \mathcal{S}} P(s,\hat{a},s')$ is the normalizing factor. As $-\log Z$ is always positive, we obtain the following lower bound,

$$\begin{aligned}
-\mathcal{L}(\hat{\phi}, \varphi) &\geq \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s,a,s') \log P(s,\hat{a},s') - C_1 \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s,a,s') \log P(\hat{a}|s,s') P(s,s') - C_1 \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}_k} \sum_{s' \in \mathcal{S}} P(s,a,s') \log P(\hat{a}|s,s') + C_2 - C_1,
\end{aligned} \tag{17}$$

where C_2 is another constant consisting of $\log P(s, s')$ and is independent of $\hat{a}$.

Now, let us focus on $P(\hat{a}|s, s')$, which represent the probability of the action $\hat{a}$ given the transition s, s'. Notice that $\hat{a}$ is selected by $\hat{\phi}$ only using $\hat{e}$. Therefore, given $\hat{e}$, probability of $\hat{a}$ is independent of everything else,

$$\log P(\hat{a}|s, s') = \log \int P(\hat{a}|\hat{e}, s, s')P(\hat{e}|s, s')\mathrm{d}\hat{e} = \log \int P(\hat{a}|\hat{e})P(\hat{e}|s, s')\mathrm{d}\hat{e}. \tag{18}$$

Let $Q(\hat{e}|s, s')$ be a parameterized distribution that encodes the context (s, s') into the structure $\hat{e}$, then, we can write (18) as,

$$\begin{aligned}
\log P(\hat{a}|s, s') &= \log \int \frac{Q(\hat{e}|s, s')}{Q(\hat{e}|s, s')} P(\hat{a}|\hat{e})P(\hat{e}|s, s')\mathrm{d}\hat{e} \\
&= \log \mathbf{E}\left[\frac{P(\hat{a}|\hat{e})P(\hat{e}|s, s')}{Q(\hat{e}|s, s')} \middle| Q(\hat{e}|s, s') \right] \\
&\geq \mathbf{E}\left[\log \frac{P(\hat{a}|\hat{e})P(\hat{e}|s, s')}{Q(\hat{e}|s, s')} \middle| Q(\hat{e}|s, s') \right] \qquad \text{(from Jensen's inequality)} \\
&= \mathbf{E}\Big[\log P(\hat{a}|\hat{e}) \Big| Q(\hat{e}|s, s') \Big] + \mathbf{E}\left[\log \frac{P(\hat{e}|s, s')}{Q(\hat{e}|s, s')} \middle| Q(\hat{e}|s, s') \right] \\
&= \mathbf{E}\Big[\log P(\hat{a}|\hat{e}) \Big| Q(\hat{e}|s, s') \Big] - \mathrm{KL}\Big(Q(\hat{e}|s, s') \Big\| P(\hat{e}|s, s') \Big).
\end{aligned} \tag{19}$$

Notice that $P(\hat{a}|e)$ and $Q(e|s, s')$ correspond to $\hat{\phi}$ and φ, respectively. $P(\hat{e}|s, s')$ corresponds to the prior on $\hat{e}$. Therefore, combining (17) and (19) we get,

$$-\mathcal{L}(\hat{\phi}, \varphi) \geq \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}_k} \sum_{s'\in\mathcal{S}} P(s, a, s') \left(\mathbf{E}\left[\log \hat{\phi}(\hat{a}|\hat{e}) \middle| \varphi(\hat{e}|s, s') \right] - \mathrm{KL}\Big(\varphi(\hat{e}|s, s') \Big\| P(\hat{e}|s, s') \Big) \right) + C,$$

where C denotes all the constants.

D: Lifelong Adaptation and Improvement

Property 2. *(Lifelong Adaptation and Improvement) In an L-MDP, let Δ denote the difference of performance between v^{μ^*} and the best achievable using our policy parameterization, then the overall sub-optimality can be expressed as,*

$$v^{\mu^*}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) = \underbrace{\sum_{k=1}^{\infty} \left(v^{\beta_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) - v^{\beta_k\hat{\phi}_k}_{\mathcal{M}_k}(s_0) \right)}_{\textit{Adaptation}} + \underbrace{\sum_{k=1}^{\infty} \left(v^{\beta^*_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) - v^{\beta_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) \right)}_{\textit{Policy Improvement}} + \Delta,$$

where $\mathcal{M}_k$ is used in the subscript to emphasize the respective L-MDP.

Proof.

$$\begin{aligned}
v^{\mu^*}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) &= v^{\mu^*}(s_0) \pm v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) \\
&= \left(v^{\mu^*}(s_0) - v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) \right) + \left(v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) \right) \\
&= \left(v^{\mu^*}(s_0) \pm v^{\beta^*_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) \right) + \left(v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) \right) \\
&= \left(v^{\mu^*}(s_0) - v^{\beta^*_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) \right) + \left(v^{\beta^*_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) \right) + \left(v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) \right).
\end{aligned}$$

As $\beta_2 \coloneqq \beta^*_1$ and $\hat{\phi}_2 \coloneqq \hat{\phi}^*_1$ in $\mathcal{M}_2$,

$$v^{\mu^*}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) = \left(v^{\mu^*}(s_0) - v^{\beta_2\hat{\phi}_2}_{\mathcal{M}_2}(s_0) \right) + \left(v^{\beta^*_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) \right) + \left(v^{\beta_1\hat{\phi}^*_1}_{\mathcal{M}_1}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) \right).$$

Notice that we have expressed the sub-optimality in $\mathcal{M}_1$ as sub-optimality in $\mathcal{M}_2$, plus adaptation and a policy improvement terms in $\mathcal{M}_1$. Expanding it one more time,

$$\begin{aligned}
v^{\mu^*}(s_0) - v^{\beta_1\hat{\phi}_1}_{\mathcal{M}_1}(s_0) = &\left(v^{\mu^*}(s_0) - v^{\beta_3\hat{\phi}_3}_{\mathcal{M}_3}(s_0) \right) + \\
&\sum_{k=1}^{2} \left(v^{\beta^*_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) - v^{\beta_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) \right) + \sum_{k=1}^{2} \left(v^{\beta_k\hat{\phi}^*_k}_{\mathcal{M}_k}(s_0) - v^{\beta_k\hat{\phi}_k}_{\mathcal{M}_k}(s_0) \right).
\end{aligned}$$

It is now straightforward to observe the result by successively 'unravelling' the sub-optimality in $\mathcal{M}_3$ in a similar fashion. The final difference between v^{μ^*} and the best policy using our proposed parameterization is Δ. □

E: Algorithm Details

A step-by-step pseudo-code for the LAICA algorithm is available in Algorithm 1. The crux of the algorithm is based on the iterative adapt and improve procedure obtained from Property 2.

We begin by initializing the parameters for $\beta_0^*, \hat{\phi}_0^*$ and φ_0^*. In Lines 3 to 5, for every change in the set of available actions, instead of re-initializing from scratch, the previous best estimates for $\beta, \hat{\phi}$ and φ are carried forward to build upon existing knowledge. As β and φ are invariant to the cardinality of the available set of actions, no new parameters are required for them. In Line 6 we add new parameters in the function $\hat{\phi}$ to deal with the new set of available actions.

To minimize the adaptation drop, we make use of Property 1. Let $\mathcal{L}^{\text{lb}}$ denote the lower bound for $\mathcal{L}$, such that,

$$\mathcal{L}^{\text{lb}}(\hat{\phi}, \varphi) \coloneqq \mathbf{E}\left[\log \hat{\phi}(\hat{A}_t|\hat{E}_t)\middle|\varphi(\hat{E}_t|S_t, S_{t+1})\right] - \lambda \text{KL}\left(\varphi(\hat{E}_t|S_t, S_{t+1})\middle\|P(\hat{E}_t|S_t, S_{t+1})\right).$$

Note that following the literature on variational auto-encoders, we have generalized (4) to use a Lagrangian λ to weight the importance of KL divergence penalty (Higgins et al. 2017).[2] When $\lambda = 1$, it degenrates to (4). We set the prior $P(\hat{e}|s, s')$ to be an isotropic normal distribution, which also allows KL to be computed in closed form (Kingma and Welling 2013). From Line 7 to 11 in the Algorithm 1, random actions from the available set of actions are executed and their corresponding transitions are collected in a buffer. Samples from this buffer are then used to maximize the lower bound objective $\mathcal{L}^{\text{lb}}$ and adapt the parameters of $\hat{\phi}$ and φ. The optimized $\hat{\phi}^*$ is then kept fixed during policy improvement.

Lines 16-22 correspond to the standard policy gradient approach for improving the performance of a policy. In our case, the policy β first outputs a vector $\hat{e}$ which gets mapped by $\hat{\phi}^*$ to an action. The observed transition is then used to compute the policy gradient (Sutton et al. 2000) for updating the parameters of β towards β^*. If a critic is used for computing the policy gradients, then it is also subsequently updated by minimizing the TD error (Sutton and Barto 2018). This iterative process of adaption and policy improvement continues for every change in the action set size.

Algorithm 1: Lifelong Adaptation and Improvement for Changing Actions (LAICA)

1 **Initialize** $\beta_0^*, \hat{\phi}_0^*, \varphi_0^*$.
2 **for** *change* $k = 1, 2...$ **do**
3 $\quad \beta_k \leftarrow \beta_{k-1}^*$
4 $\quad \varphi_k \leftarrow \varphi_{k-1}^*$
5 $\quad \hat{\phi}_k \leftarrow \hat{\phi}_{k-1}^*$
6 $\quad$ Add parameters in $\hat{\phi}_k$ for new actions
} Reuse past knowledge.
7 $\quad$ Buffer $\mathbb{B} = \{\}$
8 $\quad$ **for** $episode = 0, 1, 2...$ **do**
9 $\quad\quad$ **for** $t = 0, 1, 2...$ **do**
10 $\quad\quad\quad$ Execute random a_t and observe s_{t+1}
11 $\quad\quad\quad$ Add transition to $\mathbb{B}$
12 $\quad$ **for** $iteration = 0, 1, 2...$ **do**
13 $\quad\quad$ Sample batch $b \sim \mathbb{B}$
14 $\quad\quad$ Update $\hat{\phi}_k$ and φ_k by maximizing $\mathcal{L}^{\text{lb}}(\hat{\phi}_k, \varphi_k)$ for b
15
} Adapt $\hat{\phi}_k$ to $\hat{\phi}_k^*$
16 $\quad$ **for** $episode = 0, 1, 2...$ **do**
17 $\quad\quad$ **for** $t = 0, 1, 2...$ **do**
18 $\quad\quad\quad$ Sample $\hat{e}_t \sim \beta_k(\cdot|s_t)$
19 $\quad\quad\quad$ Map $\hat{e}_t$ to an action a_t using $\hat{\phi}_k^*(e)$
20 $\quad\quad\quad$ Execute a_t and observe s_{t+1}, r_t
21 $\quad\quad\quad$ Update β_k using any policy gradient algorithm
22 $\quad\quad\quad$ Update critic by minimizing TD error.
} Improve β_k to β_k^*

[2] Conventionally, the Lagrangian in VAE setting is denoted using β (Higgins et al. 2017). In our paper, to avoid symbol overload, we use λ for it.

F: Empirical Analysis Details

Domains

To demonstrate the effectiveness of our proposed method(s) on lifelong learning problems, we consider a maze environment and two domains corresponding to real-world applications, all with large set of changing actions. For each of these domains, the total number of actions were randomly split into five mutually exclusive sets of equal sizes. Initially, the agent only had the actions available in the first set and after every change the next set of action was made available additionally. For all our experiments, changes to the action set were made after equal intervals.

Maze. As a proof-of-concept, we constructed a continuous-state maze environment where the state comprised of the coordinates of the agent's current location. The agent has 8 equally spaced actuators (each actuator moves the agent in the direction the actuator is pointing towards) around it, and it can choose whether each actuator should be on or off. Therefore, the total number of possible actions is $2^8 = 256$. The net outcome of an action is the vectorial summation of the displacements associated with the selected actuators. The agent is penalized at each time step to encourage it to reach the goal as quickly as possible. A goal reward is given when it reaches the goal position To make the problem more challenging, random noise was added to the action 10% of the time and the maximum episode length was 150 steps.

Case Study: Real-world recommender systems. We consider two real-world applications of recommender systems that require decision making over multiple time steps and where the number of possible decisions can vary over the lifetime of the system.

First, a web-based video-tutorial platform, which has a recommendation engine that suggests a series of tutorial videos on various software. On this tutorial platform, there is a large pool of available tutorial videos on several software and new videos are uploaded periodically. This requires the recommender system to keep adjusting to these changes constantly. The aim for the recommender system is to suggest tutorials so as to meaningfully engage the user on how to use these software and convert novice users into experts in their respective areas of interest.

The second application is a professional multi-media editing software. Modern multimedia editing software often contain many tools that can be used to manipulate the media, and this wealth of options can be overwhelming for users. Further, with every major update to the software, new tools are developed and incorporated into the software to enhance user experience. In this domain, an agent suggests which of the available tools the user may want to use next. The objective is to increase user productivity and assist in achieving their end goal.

For both of these applications, an existing log of user's click stream data was used to create an n-gram based MDP model for user behavior (Shani, Heckerman, and Brafman 2005). In the tutorial recommendation task, sequences of user interaction were aggregated to obtain over 29 million clicks. Similarly, sequential usage patterns of the tools in the multi-media editing software were collected to obtain a total of over 1.75 billion user clicks. Tutorials and tools that had less than 100 clicks in total were discarded. The remaining 1498 tutorials and 1843 tools for the web-based tutorial platform and the multi-media software, respectively, corresponds to the total number of actions. The MDP had continuous state-space, where each state consisted of the feature descriptors associated with each item (tutorial or tool) in the current n-gram. Rewards were chosen based on a surrogate measure for difficulty level of tutorials and popularity of final outcomes of user interactions in the multi-media editing software, respectively.

Implementation Details

For the maze domain, single layer neural networks were used to parameterize both the actor and critic. The learning rates for policy were searched over the range $[1e-2, 1e-4]$ and for critic it was searched over the range $[5e-2, 5e-4]$. State features were represented using the 3$^{\text{rd}}$ order coupled Fourier basis (Konidaris, Osentoski, and Thomas 2011). The discounting parameter γ was set to 0.99 and eligibility traces to 0.9. Since it was a toy domain, the output dimension of β was kept fixed to 2. After every change in the action set, 500 randomly drawn trajectories were used to update $\hat{\phi}$. The value of λ was searched over the range $[1e-2, 1e-4]$.

For the real-world environments, 2 layer neural networks were used to parameterize both the actor and critic. The learning rates for both were searched over the range $[1e-2, 1e-4]$. Similar to prior works, the module for encoding state features was shared to reduce the number of parameters, and the learning rate for it was additionally searched over $[1e-2, 1e-4]$. The dimension of the neural network's hidden layer was searched over $\{64, 128, 256\}$. The discounting parameter γ was set to 0.9. For actor-critic based results eligibility traces was set to 0.9 and for DPG the target actor and policy update rate was fixed to its default setting of 0.001. The output dimension of β was searched over $\{16, 32, 64\}$. After every change in the action set, samples from 2000 randomly drawn trajectories were used to update $\hat{\phi}$.

For all the results of the LAICA, since the output of β was defined over a continuous space, it was parameterized as the isotropic normal distribution. The value for variance was kept fix for the Maze domain and was searched over $[0.5, 1.5]$. For the real-world domains, the variance was parameterized and learned along with other parameters. The function φ was parameterized to concatenate the state features of both s and s' and use a single layer neural network to project to a space corresponding to

the inferred structure in the actions. The function $\hat{\phi}$ was linearly parameterized to compute a Boltzmann distribution over the available set of actions. After every change in the action set, new rows were stacked in its weight matrix for generating scores for the new actions. The learning rates for functions $\hat{\phi}$ and φ were jointly searched over $[1e-2, 1e-4]$.

As our proposed method decomposes the overall policy into two components, the resulting architecture resembles that of a one layer deeper neural network. Therefore, for the baselines, we ran the experiments with a hyper-parameter search for policies with additional depths $\{1, 2, 3\}$, each with different combinations of width $\{2, 16, 64\}$. The remaining architectural aspects and properties of the hyper-parameter search for the baselines were performed in the same way as mentioned above for our proposed method. For dealing with new actions, new rows were stacked in the weight matrix of the last layer of the policy in Baseline(2).

In total, 200 settings for each algorithm, for each domain, were uniformly sampled from the respective hyper-parameter ranges/sets mentioned. Results from the best performing setting are reported in all the plots. Each hyper-parameter setting was independently ran using 10 different seeds to get the standard error of the performance.