Sequential Transfer in Reinforcement Learning with a Generative Model

Andrea Tirinzoni [1] **Riccardo Poiani** [1] **Marcello Restelli** [1]

Abstract

We are interested in how to design reinforcement learning agents that provably reduce the sample complexity for learning new tasks by transferring knowledge from previously-solved ones. The availability of solutions to related problems poses a fundamental trade-off: whether to seek policies that are expected to achieve high (yet suboptimal) performance in the new task immediately or whether to seek information to quickly identify an optimal solution, potentially at the cost of poor initial behavior. In this work, we focus on the second objective when the agent has access to a generative model of state-action pairs. First, given a set of solved tasks containing an approximation of the target one, we design an algorithm that quickly identifies an accurate solution by seeking the state-action pairs that are most informative for this purpose. We derive PAC bounds on its sample complexity which clearly demonstrate the benefits of using this kind of prior knowledge. Then, we show how to learn these approximate tasks sequentially by reducing our transfer setting to a hidden Markov model and employing spectral methods to recover its parameters. Finally, we empirically verify our theoretical findings in simple simulated domains.

1. Introduction

Knowledge transfer has proven to be a fundamental tool for enabling lifelong reinforcement learning (RL) (Sutton & Barto, 1998), where agents face sequences of related tasks. In this context, upon receiving a new task, the agent aims at reusing knowledge from the previously-solved ones to speed-up the learning process. This has the potential to achieve significant performance improvements over standard RL from scratch, and several studies have confirmed this hypothesis (Taylor & Stone, 2009; Lazaric, 2012).

[1]Politecnico di Milano, Milan, Italy. Correspondence to: Andrea Tirinzoni <andrea.tirinzoni@polimi.it>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

A key question is what and how knowledge should be transferred (Taylor & Stone, 2009). As for the kind of knowledge to be reused, a variety of algorithms have been proposed to transfer experience samples (Lazaric et al., 2008; Taylor et al., 2008; Yin & Pan, 2017; Tirinzoni et al., 2018b; 2019), policies (Fernández & Veloso, 2006; Mahmud et al., 2013; Rosman et al., 2016; Abel et al., 2018; Yang et al., 2019; Feng et al., 2019), value-functions (Liu & Stone, 2006; Tirinzoni et al., 2018a), features (Konidaris et al., 2012; Barreto et al., 2017; 2018), and more. Regarding how knowledge should be transferred, the answer is more subtle and depends directly on the desired objectives. On the one hand, one could aim at maximizing the *jumpstart*, i.e., the expected initial performance in the target task. This can be done by transferring *initializers*, e.g., policies or value-functions that, based on past knowledge, are expected to immediately yield high rewards. Among the theoretical studies, Mann & Choe (2012) and Abel et al. (2018) showed that jumpstart policies can provably reduce the number of samples needed for learning a target task. However, since these initializers are computed before receiving the new task, the agent, though starting from good performance, might still take a long time before converging to near-optimal behavior. An alternative approach is to spend some initial interactions with the target to gather information about the task itself, so as to better decide what to transfer. For instance, the agent could aim at identifying which of the previously-solved problems is most similar to the target. If the similarity is actually large, transferring the past solution would instantaneously lead to near-optimal behavior. This *task identification* problem has been studied by Dyagilev et al. (2008); Brunskill & Li (2013); Liu et al. (2016). One downside is that these approaches do not actively seek information to minimize identification time (or, equivalently, sample complexity), in part because it is non-trivial to find which actions are the most informative given knowledge from related tasks. To the best of our knowledge, it is an open problem how to leverage this prior knowledge to reduce identification time.

Regardless of how knowledge is transferred, a common assumption is that tasks are drawn from some fixed distribution (Taylor & Stone, 2009), which often hides the sequential nature of the problem. In many lifelong learning problems, tasks evolve dynamically and are temporally correlated. Consider an autonomous car driving through traffic.

Based on the traffic conditions, both the environment dynamics and the agent's desiderata could significantly change. However, this change follows certain temporal patterns, e.g., the traffic level might be very high in the early morning, medium in the afternoon, and so on. This setting could thus be modeled as a sequence of related RL problems with temporal dependencies. It is therefore natural to expect that a good transfer agent will exploit the sequential structure among tasks rather than only their static similarities.

Our paper aims at advancing the theory of transfer in RL by addressing the following questions related to the above-mentioned points: (1) how can the agent quickly identify an accurate solution of the target task when provided with solved related problems? (2) how can the agent exploit the sequential nature of the lifelong learning setup?

Taking inspiration from previous works on non-stationary RL (Choi et al., 2000a; Hadoux et al., 2014), we model the sequential transfer setting by assuming that the agent faces a finite number of tasks whose evolution is governed by an underlying Markov chain. Each task has potentially different rewards and dynamics, while the state-action space is fixed and shared. The motivation for this framework is that some real systems often work in a small number of operating *modes* (e.g., the traffic condition in the example above), each with different dynamics. Unlike these works, we assume that the agent is informed when a new task arrives as in the standard transfer setting (Taylor & Stone, 2009). Then, as in previous studies on transfer in RL (Brunskill & Li, 2013; Azar et al., 2013a), we decompose the problem into two parts. First, in Section 3, we assume that the agent is provided with prior knowledge in the form of a set of tasks containing an approximation to the target one. Under the assumption that a generative model of the environment is available, we design an algorithm that actively seeks information in order to identify a near-optimal policy and transfers it to the target task. We derive PAC bounds (Strehl et al., 2009) on its sample complexity which are significantly tighter than existing results and clearly demonstrate performance improvements over learning from scratch. Then, in Section 4, we show how this prior knowledge can be learned in our sequential setup to allow knowledge transfer. We reduce the problem to learning a hidden Markov model and use spectral methods (Anandkumar et al., 2014) to estimate the task models necessary for running our policy transfer algorithm. We derive finite-sample bounds on the error of these estimated models and discuss how to leverage the temporal correlations to further reduce the sample complexity of our methods. Finally, we report numerical simulations in some standard RL benchmarks which confirm our theoretical findings. In particular, we show examples where identification yields faster convergence than jumpstart methods, and examples where the exploitation of the sequential correlations among tasks is superior than neglecting them.

2. Preliminaries

We model each task θ, as a discounted Markov decision process (MDP) (Puterman, 2014), $\mathcal{M}_\theta := \langle \mathcal{S}, \mathcal{A}, p_\theta, q_\theta, \gamma \rangle$, where $\mathcal{S}$ is a finite set of S states, $\mathcal{A}$ is a finite set of A actions, $p_\theta : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ are the transition probabilities, $q_\theta : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{U})$ is the reward distribution over space $\mathcal{U}$ (e.g., $\mathcal{U} = \mathbb{R}$), and $\gamma \in [0, 1)$ is a discount factor. Here $\mathcal{P}(\Omega)$ denotes the set of probability distributions over a set Ω. We suppose that the sets of states and actions are fixed and that tasks are uniquely determined by their parameter θ. For this reason, we shall occasionally use θ to indicate $\mathcal{M}_\theta$, while alternatively referring to it as task, parameter, or (MDP) model. We denote by $r_\theta(s, a) := \mathbb{E}_{q_\theta}[U_t | S_t = s, A_t = a]$ and assume, without loss of generality, that rewards take values in $[0, 1]$. We use $V_\theta^\pi(s)$ to indicate the value function of a (deterministic) policy $\pi : \mathcal{S} \to \mathcal{A}$ in task θ, i.e., the expected discounted return achieved by π when starting from state s in θ, $V_\theta^\pi(s) := \mathbb{E}_\theta^\pi \left[\sum_{t=0}^\infty \gamma^t U_t | S_0 = s\right]$. The optimal policy π_θ^* for task θ maximizes the value function at all states simultaneously, $V_\theta^{\pi_\theta^*}(s) \geq V_\theta^\pi(s)$ for all s and π. We let $V_\theta^*(s) := V_\theta^{\pi_\theta^*}(s)$ denote the optimal value function of θ. Given $\epsilon > 0$, we say that a policy π is ϵ-optimal for θ if $V_\theta^\pi(s) \geq V_\theta^*(s) - \epsilon$ for all states s. We define $\sigma_\theta^r(s, a)^2 := \mathbb{V}\mathrm{ar}_{q_\theta(\cdot|s,a)}[U]$ as the variance of the reward in s, a for task θ, and $\sigma_\theta^p(s, a; \theta')^2 := \mathbb{V}\mathrm{ar}_{p_\theta(\cdot|s,a)}[V_{\theta'}^*(S')]$ as the variance of the optimal value function of θ' under the transition model of θ. To simplify the exposition, we shall alternatively use the standard vector notation to indicate these quantities. For instance, $V_\theta^* \in \mathbb{R}^S$ is the vector of optimal values, $p_\theta(s, a) \in \mathbb{R}^S$ is the vector of transition probabilities from s, a, $r_\theta \in \mathbb{R}^{SA}$ is the flattened reward matrix, and so on. To measure the distance between two models θ, θ', we define $\Delta_{s,a}^r(\theta, \theta') := |r_\theta(s, a) - r_{\theta'}(s, a)|$ for the rewards and $\Delta_{s,a}^p(\theta, \theta') = |(p_\theta(s, a) - p_{\theta'}(s, a))^T V_\theta^*|$ for the transition probabilities. The latter measures how much the expected return of an agent taking s, a and acting optimally in θ changes when the first transition only is governed by θ'. See Appendix A for a quick reference of notation.

Sequential Transfer Setting We model our problem as a hidden-mode MDP (Choi et al., 2000b). We suppose that the agent faces a finite number of k possible tasks, $\Theta = \{\theta_1, \theta_2, \dots, \theta_k\}$. Tasks arrive in sequence and evolve according to a Markov chain with transition matrix $T \in [0, 1]^{k \times k}$ and an arbitrary initial task distribution. Let θ_h^*, for $h = 1, \dots, m$, be the h-th random task faced by the agent. Then, $[T]_{i,j} = \mathbb{P}\left\{\theta_{h+1}^* = \theta_i | \theta_h^* = \theta_j\right\}$. The agent only knows the number of tasks k, while both the MDP models and the task-transition matrix T are unknown. The goal is to identify an ϵ-optimal policy for θ_h^* using as few samples as possible, while leveraging knowledge from the previous tasks $\theta_1^*, \dots, \theta_{h-1}^*$ to further reduce the sample complexity. In order to provide deeper insights into how

to design efficient identification strategies and facilitate the analysis, we assume that the agent can access a generative model of state-action pairs. Similarly to experimental optimal design (Pukelsheim, 2006), upon receiving each new task θ^*_h, the agent can perform at most n *experiments* for identification, where each experiment consists in choosing an arbitrary state-action pair s, a and receiving a random next-state $S' \sim p_{\theta^*_h}(\cdot|s,a)$ and reward $R \sim q_{\theta^*_h}(\cdot|s,a)$. After this *identification phase*, the agent has to transfer a policy to the target task and starts interacting with the environment in the standard online fashion.

As in prior works (e.g., Brunskill & Li, 2013; Azar et al., 2013a; Liu et al., 2016), we suppose that the agent maintains an approximation to all MDP models, $\{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$, and decompose the problem in two main steps.[1] (1) First, in Section 3 we present an algorithm that, given the approximate models together with the corresponding approximation errors, actively queries the generative model of the target task by seeking state-action pairs that yield high information about the optimal policy of θ^*_h. (2) Then, in Section 4 we show how the agent can build these approximate models from the samples collected while interacting with the sequence of tasks so far.

3. Policy Transfer from Uncertain Models

Throughout this section, we shall focus on learning a single model $\theta^* \in \Theta$ given knowledge from previous tasks. Therefore, to simplify the notation, we shall drop dependency on the specific task sequence (i.e., we drop the task indexes h).

Let $\{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$ be the approximate MDP models, with $\widetilde{p}_\theta$ and $\widetilde{r}_\theta$ denoting their transition probabilities and rewards, respectively. We use the intuition that, if these approximations are accurate enough, the problem can be reduced to identifying a suitable policy among the optimal ones of the MDPs $\{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$, which necessarily contains an approximation of $\pi^*_{\theta^*}$. We rely on the following assumption to assess the quality of the approximate models.

Assumption 1 (Model uncertainty)**.** *There exist known constants* $\Delta^r_{\max}$, $\Delta^p_{\max}$, $\Delta^{\sigma_r}_{\max}$, $\Delta^{\sigma_p}_{\max}$ *such that*

$$\max_{\theta\in\Theta} \|r_\theta - \widetilde{r}_\theta\|_\infty \le \Delta^r_{\max},$$
$$\max_{\theta,\theta'\in\Theta} \|(p_\theta - \widetilde{p}_\theta)^T \widetilde{V}^*_{\theta'}\|_\infty \le \Delta^p_{\max},$$
$$\max_{\theta\in\Theta} \|\sigma^r_\theta - \widetilde{\sigma}^r_\theta\|_\infty \le \Delta^{\sigma_r}_{\max},$$
$$\max_{\theta,\theta'\in\Theta} \|\sigma^p_\theta(\theta') - \widetilde{\sigma}^p_\theta(\theta')\|_\infty \le \Delta^{\sigma_p}_{\max}.$$

Assumption 1 ensures that an upper bound to the approximation error of the given models is known. In particular, the maximum error among all components, $\Delta := \max\{\Delta^r_{\max}, \Delta^p_{\max}, \Delta^{\sigma_r}_{\max}, \Delta^{\sigma_p}_{\max}\}$ is a fundamental parameter for our approach. In Section 4, we shall see how to guarantee this assumption when facing tasks sequentially.

[1] In the remainder, we overload the notation by using tildes to indicate quantities related to approximate models.

Algorithm 1 Policy Transfer from Uncertain Models (PTUM)

Require: Set of approximate MDPs $\{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$, accuracy ϵ, confidence δ, number of samples n, model uncertainty Δ
Ensure: An ϵ-optimal policy for $\mathcal{M}_{\theta^*}$ with probability $1-\delta$

1: // CHECK ACCURACY CONDITION
2: If $\Delta \ge \frac{\epsilon(1-\gamma)}{4(1+\gamma)} \to$ Stop and run (ϵ,δ)-PAC algorithm
3: // TRANSFER MODE
4: Initialize datasets $\mathcal{D}^r_{s,a}, \mathcal{D}^p_{s,a} \leftarrow \emptyset$
5: **for** $t = 1, 2, \dots, n$ **do**
6: // STEP 1. BUILD EMPIRICAL MDP MODEL
7: $\widehat{r}_t(s,a) \leftarrow \frac{1}{N_t(s,a)} \sum_{u\in\mathcal{D}^r_{s,a}} u$
8: $\widehat{p}_t(s'|s,a) \leftarrow \frac{1}{N_t(s,a)} \sum_{s''\in\mathcal{D}^p_{s,a}} \mathbb{1}\{s' = s''\}$
9: $\widehat{\sigma}^r_t(s,a)^2 \leftarrow \frac{\sum_{u\in\mathcal{D}^r_{s,a}} (u - \widehat{r}_t(s,a))^2}{N_t(s,a)-1}$
10: $\widehat{\sigma}^p_t(s,a;\theta')^2 \leftarrow \frac{\sum_{s''\in\mathcal{D}^p_{s,a}} (\widetilde{V}^*_{\theta'}(s'') - \widehat{p}_t(s,a)^T \widetilde{V}^*_{\theta'})^2}{N_t(s,a)-1}$
11: // STEP 2. UPDATE CONFIDENCE SET
12: $\bar{\Theta}_t \leftarrow \left\{\theta \in \bar{\Theta}_{t-1} \;\middle|\; (1)\text{-}(4) \text{ hold for all } s,a \text{ and } \theta' \in \Theta\right\}$
13: // STEP 3. CHECK STOPPING CONDITION
14: If there exists $\theta \in \bar{\Theta}_t$ such that for all $\theta' \in \bar{\Theta}_t$ we have $\widetilde{V}^{\widetilde{\pi}^*_\theta}_{\theta'} \ge \widetilde{V}^*_{\theta'} - \epsilon + \frac{2\Delta(1+\gamma)}{1-\gamma} \to$ Stop and return $\widetilde{\pi}^*_\theta$
15: // STEP 4. QUERY GENERATIVE MODEL
16: $\mathcal{I}^r_t(s,a) \leftarrow \max_{\theta,\theta'\in\bar{\Theta}_t} \mathcal{I}^r_{s,a}(\theta,\theta')$
17: $\mathcal{I}^p_t(s,a) \leftarrow \max_{\theta,\theta'\in\bar{\Theta}_t} \mathcal{I}^p_{s,a}(\theta,\theta')$
18: $(S_t, A_t) \leftarrow \operatorname{argmax}_{s,a} \max\{\mathcal{I}^r_t(s,a), \mathcal{I}^p_t(s,a)\}$
19: Obtain $S_{t+1} \sim p_{\theta^*}(\cdot|S_t,A_t)$ and $U_{t+1} \sim q_{\theta^*}(\cdot|S_t,A_t)$
20: Store $\mathcal{D}^p_{s,a} = \mathcal{D}^p_{s,a} \cup \{S_{t+1}\}$ and $\mathcal{D}^r_{s,a} = \mathcal{D}^r_{s,a} \cup \{U_{t+1}\}$
21: **end for**
22: If the algorithm did not stop $\to$ Run (ϵ,δ)-PAC algorithm

3.1. The PTUM Algorithm

We now present Policy Transfer from Uncertain Models (PTUM), whose pseudo-code is provided in Algorithm 1. Given the approximate models $\{\widetilde{\mathcal{M}}_\theta\}_{\theta\in\Theta}$, whose maximum error is bounded by Δ from Assumption 1, and two values $\epsilon, \delta > 0$, our approach returns a policy which, with probability at least $1-\delta$, is ϵ-optimal for the target task θ^*. We now describe in detail all the steps of Algorithm 1.

First (lines 1-2), we check whether positive transfer can occur. The intuition is that, if the approximate models are too uncertain (i.e., Δ is large), the transfer of a policy might actually lead to poor performance in the target task, i.e., *negative transfer* occurs. Our algorithm checks whether Δ is below a certain threshold (line 2). Otherwise, we run any (ϵ,δ)-PAC[2] algorithm to obtain an ϵ-optimal policy. Later on, we shall discuss how this algorithm could be

[2] An algorithm is (ϵ,δ)-PAC if, with probability $1-\delta$, it computes an ϵ-optimal policy using a polynomial number of samples in the relevant problem-dependent quantities (Strehl et al., 2009).

chosen. Although the condition at line 2 seems restrictive, as Δ is required to be below a factor of ϵ, we conjecture this dependency to be nearly-tight (at least in a worst-case sense). In fact, Feng et al. (2019) have recently shown that the sole knowledge of a poorly-approximate model cannot reduce the worst-case sample complexity of any agent seeking an ϵ-optimal policy. If the condition at line 2 fails, i.e., the models are accurate enough, we say that the algorithm enters the *transfer mode*. Here, the generative model is queried online until an ϵ-optimal policy is found. Similarly to existing works on model identification (Dyagilev et al., 2008; Brunskill & Li, 2013), the algorithm proceeds by elimination. At each time-step t (up to at most n), we keep a set $\bar{\Theta}_t \subseteq \Theta$ of those models, called *active*, that are likely to be (close approximations of) the target θ^*. This set is created based on the distance between each model and the empirical MDP constructed with the observed samples. Then, the algorithm chooses the next state-action pair S_t, A_t to query the generative model so that the samples from S_t, A_t are informative to eliminate one of the "wrong" models from the active set. This process is iterated until the algorithm finds a policy that is ϵ-optimal for all active models, in which case the algorithm stops and returns such policy. We now describe these main steps in detail.

Step 1. Building the empirical MDP In order to find the set of active models $\bar{\Theta}_t$ at time t, the algorithm first builds an empirical MDP as a proxy for the true one. Let $N_t(s,a)$ be the number of samples collected from s,a up to (and not including) time t. First, the algorithm estimates, for each s,a, the empirical rewards $\widehat{r}_t(s,a)$ and transition probabilities $\widehat{p}_t(s,a)$ (lines 7-8). Then, it computes the empirical variance of the rewards $\widehat{\sigma}^r_t(s,a)^2$ and of the optimal value functions $\widehat{\sigma}^p_t(s,a;\theta')^2$ for each model $\theta' \in \Theta$ (lines 9-10). This quantities are arbitrarily initialized when $N_t(s,a) = 0$.

Step 2. Building the confidence set We define the confidence set $\bar{\Theta}_t$ as the set of models that are "compatible" with the empirical MDP in *all* steps up to t. Formally, a model $\theta \in \Theta$ belongs to the confidence set $\bar{\Theta}_t$ at time t if it was active before (i.e., $\theta \in \bar{\Theta}_{t-1}$) and the following conditions are satisfied for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $\theta' \in \Theta$:

$$|\widehat{r}_t(s,a) - \widetilde{r}_\theta(s,a)| \le C^r_{t,\delta}(s,a), \tag{1}$$

$$|(\widehat{p}_t(s,a) - \widetilde{p}_\theta(s,a))^\intercal \widetilde{V}^*_{\theta'}| \le C^p_{t,\delta}(s,a,\theta'), \tag{2}$$

$$|\widehat{\sigma}^r_t(s,a) - \widetilde{\sigma}^r_\theta(s,a)| \le C^{\sigma_r}_{t,\delta}(s,a), \tag{3}$$

$$|\widehat{\sigma}^p_t(s,a;\theta') - \widetilde{\sigma}^p_\theta(s,a;\theta')| \le C^{\sigma_p}_{t,\delta}(s,a). \tag{4}$$

Intuitively, a model belongs to the confidence set if its distance to the empirical MDP does not exceed, in any component, a suitable confidence level $C_{t,\delta}(s,a)$. The latter has the form $\sqrt{\frac{\log c/\delta}{N_t(s,a)}}$ and is obtained from standard applications of Bernstein's inequality (Boucheron et al., 2003). We refer the reader to Appendix B for the full expression. Alternatively, we say that a model is *eliminated* from the confidence set (i.e., it will never be active again) as soon as it is not compatible with the empirical MDP. It is important to note that, with probability at least $1-\delta$, the target task θ^* is never eliminated from $\bar{\Theta}_t$ (see Lemma 3 in Appendix B).

Step 3. Checking whether to stop After building the confidence set, the algorithm checks whether the optimal policy of some active model is $(\epsilon - 2\Delta\frac{1+\gamma}{1-\gamma})$-optimal in all other models in $\bar{\Theta}_t$, in which case it stops and returns this policy. As we shall see in our analysis, this ensures that the returned policy is also ϵ-optimal for θ^*.

Step 4. Deciding where to query the generative model The final step involves choosing the next state-action pair S_t, A_t from which to obtain a sample. This is a key point as the sampling rule is what directly determines the sample-complexity of the algorithm. As discussed previously, our algorithm eliminates the models from $\bar{\Theta}_t$ until the stopping condition is verified. Therefore, a good sampling rule should aim at minimizing the stopping time, i.e., it should aim at eliminating as soon as possible all models that prevent the algorithm from stopping. The design of our strategy is driven by this principle. Given the set of active models $\bar{\Theta}_t$, we compute, for each s,a, a score $\mathcal{I}_t(s,a)$, which we refer to as the *index* of s,a, that is directly related to the information to discriminate between any two active models using s,a only (lines 20-22). Then, we choose the s,a that maximizes the index, which can be interpreted as sampling the state-action pair that allows us to discard an active model in the shortest possible time. We confirm this intuition in our analysis later. Formally, our information measure is defined as follows.

Definition 1 (Information for model discrimination)**.** *Let $\theta, \theta' \in \Theta$. For $\widetilde{\Delta}^x_{s,a} := [\widetilde{\Delta}^x_{s,a}(\theta,\theta') - 8\Delta]_+$, with $x \in \{r,p\}$, the information for discriminating between θ and θ' using reward/transition samples from s,a are, respectively,*

$$\mathcal{I}^r_{s,a}(\theta,\theta') = \min\left\{\left(\frac{\widetilde{\Delta}^r_{s,a}}{\widetilde{\sigma}^r_\theta(s,a)}\right)^2, \widetilde{\Delta}^r_{s,a}\right\},$$

$$\mathcal{I}^p_{s,a}(\theta,\theta') = \min\left\{\left(\frac{\widetilde{\Delta}^p_{s,a}}{\widetilde{\sigma}^p_\theta(s,a;\theta)}\right)^2, (1-\gamma)\widetilde{\Delta}^p_{s,a}\right\}.$$

The total information is the maximum of these two, $\mathcal{I}_{s,a}(\theta,\theta') = \max\left\{\mathcal{I}^r_{s,a}(\theta,\theta'), \mathcal{I}^p_{s,a}(\theta,\theta')\right\}$.

The information $\mathcal{I}$ is a fundamental tool for our analysis and it can be understood as follows. The terms on the left-hand side are ratios of the squared deviation between the means of the random variables involved and their variance. If these random variables were Gaussian, this would be proportional

to the Kullback-Leibler divergence between the distributions induced by the two models, which in turn is related to their mutual information. The terms on the right-hand side arise from our choice of Bernstein's confidence intervals but have a minor role in the algorithm and its analysis.

3.2. Sample Complexity Analysis

We now analyze the sample complexity of Algorithm 1. Throughout the analysis, we assume that the model uncertainty Δ is such that Algorithm 1 enters the transfer mode and that the sample budget n is large enough to allow the identification. The opposite case is of minor importance as it reduces to the analysis of the chosen (ϵ, δ)-PAC algorithm.

Theorem 1. *Assume Δ is such that Algorithm 1 enters the transfer mode. Let τ be the random stopping time and π_τ be the returned policy. Then, with probability at least $1-\delta$, π_τ is ϵ-optimal for θ^* and the total number of queries to the generative model can be bounded by*

$$\tau \leq \frac{128 \min\{SA, |\Theta|\} \log(8SAn(|\Theta|+1)/\delta)}{\max_{s,a} \min_{\theta \in \Theta_\epsilon} \mathcal{I}_{s,a}(\theta^*, \theta)},$$

where, for $\kappa_\epsilon := \frac{(1-\gamma)\epsilon}{4} - \frac{\Delta(1+\gamma)}{2}$, the set $\Theta_\epsilon \subseteq \Theta$ is

$$\Theta_\epsilon := \left\{ \theta \;\middle|\; \|\widetilde{r}_\theta - \widetilde{r}_{\theta^*}\| > \kappa_\epsilon \vee \|(\widetilde{p}_\theta - \widetilde{p}_{\theta^*})^T \widetilde{V}^*_{\theta^*}\| > \frac{\kappa_\epsilon}{\gamma} \right\}.$$

The proof, provided in Appendix B, combines standard techniques used to analyze PAC algorithms (Azar et al., 2013b; Zanette et al., 2019) with recent ideas in field of structured multi-armed bandits (Tirinzoni et al., 2020). At first glance, Theorem 1 looks quite different from the standard sample complexity bounds available in the literature (Strehl et al., 2009). We shall see now that it reveals many interesting properties. First, this result implies that PTUM is (ϵ, δ)-PAC as the sample complexity is bounded by polynomial functions of all the relevant quantities. Next, we note that, except for logarithmic terms, the sample complexity scales with the minimum between the number of tasks and the number of state-action pairs. As in practice, we expect the former to be much smaller than the latter, we get a significant gain compared to the no-transfer case, where, even with a generative model, the sample complexity is at least linear in SA. The set Θ_ϵ can be understood as the set of all models in Θ whose optimal policy cannot be guaranteed as ϵ-optimal for the target θ^*. As Lemma 6 in Appendix B shows, it is sufficient to eliminate all models in this set to ensure stopping. Our key result is that the sample complexity of PTUM is proportional to the one of an "oracle" strategy that knows in advance the most informative state-action pairs to achieve this elimination. Note, in fact, that the denominator involves the information to discriminate any model in Θ_ϵ with θ^*, but the latter is not known to the algorithm. The following result provides further insights into the improvements over the no-transfer case.

Corollary 1. *Let Γ be the minimum gap between θ^* and any other model in Θ,*

$$\Gamma := \min_{\theta \neq \theta^*} \max \left\{ \|\widetilde{r}_\theta - \widetilde{r}_{\theta^*}\|, \|(\widetilde{p}_\theta - \widetilde{p}_{\theta^*})^T \widetilde{V}^*_{\theta^*}\| \right\}.$$

Then, with probability at least $1-\delta$,

$$\tau \leq \widetilde{\mathcal{O}} \left(\frac{\min\{SA, |\Theta|\} \log(1/\delta)}{\max\{\Gamma^2, \epsilon^2\}(1-\gamma)^4} \right).$$

This result reveals that the sample complexity of Algorithm 1 does not scale with ϵ, which is typically regarded as the main term in PAC bounds. That is, when ϵ is small, the bound scales with the minimum gap Γ between the approximate models. Interestingly, the dependence on Γ is the same as the one obtained by Brunskill & Li (2013), but in our case it constitutes a worst-case scenario since Γ can be regarded as the *minimum* positive information for model discrimination, while Theorem 1 scales with the maximum one. Moreover, since our sample complexity bound is never worse than the one of Brunskill & Li (2013) and theirs achieves robustness to negative transfer, PTUM directly inherits this property. We note that $\Gamma > 0$ since, otherwise, two identical models would exist and one could be safely neglected. The key consequence is that one could set $\epsilon = 0$ and the algorithm would retrieve an optimal policy. However, this requires the models to be perfectly approximated so as to enter the transfer mode. Finally, we point out that the optimal dependence on the discount factor was proved to be $\mathcal{O}(1/(1-\gamma)^3)$ (Azar et al., 2013b). Here, we get a slightly sub-optimal result since, for simplicity, we naively upper-bounded the variances with their maximum value, but a more involved analysis (e.g., those by Azar et al. (2013b); Sidford et al. (2018)) should lead to optimal dependence.

3.3. Discussion

The sampling procedure of PTUM (Step 4) relies on the availability of a generative model to query informative state-action pairs. We note that the definition of informative state-action pair and all other components of the algorithm are independent on the assumption that a generative model is available. Therefore, one could use ideas similar to PTUM even in the absence of a generative model. For instance, taking inspiration from E^3 (Kearns & Singh, 2002), we could build a surrogate "exploration" MDP with high rewards in informative state-action pairs and solve this MDP to obtain a policy that autonomously navigates the true environment to collect information. Alternatively, we could use the information measure $\mathcal{I}_{s,a}$ as an exploration bonus (Jian et al., 2019). We conjecture that the analysis of this kind of approaches would follow quite naturally from the one of PTUM under

the standard assumption of finite MDP diameter $D < \infty$ (Jaksch et al., 2010), for which the resulting bound would have an extra linear scaling in D as in prior works (Brunskill & Li, 2013).

PTUM calls an (ϵ, δ)-PAC algorithm whenever the models are too inaccurate or whenever n queries to the generative model are not sufficient to identify a near-optimal policy. This algorithm can be freely chosen among those available in the literature. For instance, we could choose the MaxQInit algorithm of Abel et al. (2018) which uses an optimistic value function to initialize the learning process of a PAC-MDP method (Strehl et al., 2009). In our case, the information about θ^* collected through the generative model could be used to compute much tighter upper bounds to the optimal value function than those obtained solely from previous tasks, thus significantly reducing the overall sample complexity. Alternatively, we could use the Finite-Model-RL algorithm of (Brunskill & Li, 2013) or the Parameter ELimination (PEL) method of (Dyagilev et al., 2008) by passing the set of survived models $\widetilde{\Theta}_n$ instead of Θ, so that the number of remaining eliminations is potentially much smaller than $|\Theta|$.

4. Learning Sequentially-Related Tasks

We now show how to estimate the MDP models (i.e., the reward and transition probabilities) for each θ sequentially, together with the task-transition matrix T. The method we propose allows us to obtain confidence bounds over these estimates which can be directly plugged into Algorithm 1 to find an ϵ-optimal policy for each new task.

We start by noticing that our sequential transfer setting can be formulated as a hidden Markov model (HMM) where the target tasks $\{\theta^*_h\}_{h\geq 1}$ are the unobserved variables, or hidden states, which evolve according to T, and the samples collected while learning each task are the agent's observations. Formally, consider the h-th task θ^*_h and, with some abuse of notation, let $\widehat{q}_h(u|s,a)$ and $\widehat{p}_h(s'|s,a)$ respectively denote the empirical reward distribution and transition model estimated after solving θ^*_h and, without loss of generality, after collecting at least one sample from each state-action pair. In order to simplify the exposition and analysis, we assume, in this section only, that the reward-space is finite with U elements.[3] It is easy to see that $\mathbb{E}\left[\widehat{q}_h(u|s,a)|\theta^*_h = \theta\right] = q_\theta(u|s,a)$ and $\mathbb{E}\left[\widehat{p}_h(s'|s,a)|\theta^*_h = \theta\right] = p_\theta(s'|s,a)$. Furthermore, the random variables $\widehat{q}_h(u|s,a)$ and $\widehat{p}_h(s'|s,a)$ are conditionally independent of all other variables given the target task θ^*_h. Let $o_h \in \mathbb{R}^d$, for $d = SA(S+U)$, be a vectorized version of these empirical models (i.e., our observation vector), $o_h = [\mathrm{vec}(\widehat{q}); \mathrm{vec}(\widehat{p})]$. Let $O \in \mathbb{R}^{d\times k}$ be defined as $O = [\mu_1, \mu_2, \ldots, \mu_k]$ with $\mu_j := \mathbb{E}\left[o_h|\theta^*_h = \theta_j\right]$. Intuitively, O has the (exact) flattened MDP models as its columns and can be regarded as the matrix of conditional-observation means for our HMM. Then, the problem reduces to estimating the matrices T and O while facing sequential tasks, which is a standard HMM learning problem. Spectral methods (Anandkumar et al., 2012; 2014) have been widely applied for this purpose. Besides their good empirical performance, these approaches typically come with strong finite-sample guarantees, which is a fundamental tool for our study. We also note that spectral methods have already been applied to solve problems related to ours, including transfer in multi-armed bandits (Azar et al., 2013a) and learning POMDPs (Azizzadenesheli et al., 2016; Guo et al., 2016). Here, we apply the tensor decomposition approach of Anandkumar et al. (2014). As their algorithm is quite involved, we include a brief description in Appendix C, while in the following we focus on analyzing the estimated HMM parameters.

[3] The proposed methods can be applied to continuous reward distributions, e.g., Gaussian, with only minor tweaks.

Algorithm 2 Sequential Transfer

1: Initialize set of approximate models $\{\widetilde{\mathcal{M}}^1_\theta\}_{\theta\in\Theta}$, model errors $\Delta_1 \leftarrow \infty$, and initial active set $\widetilde{\Theta}_1 \leftarrow \Theta$
2: **for** $h = 1, 2, \ldots$ **do**
3: Receive new task $\theta^*_h \sim T(\cdot|\theta^*_{h-1})$
4: Run Algorithm 1 with $\{\widetilde{\mathcal{M}}^h_\theta\}_{\theta\in\widetilde{\Theta}_h}$ and Δ_h
5: Obtain estimates $\widehat{q}_h(u|s,a)$ and $\widehat{p}_h(s'|s,a)$
6: Estimate $\widehat{O}_h, \widehat{T}_h$ using Algorithm 3 in Appendix C
7: Update model estimates $\{\widetilde{\mathcal{M}}^{h+1}_\theta\}_{\theta\in\Theta}$ from $\widehat{O}_h$
8: Compute model error Δ_h
9: Predict set of initial models $\widetilde{\Theta}_{h+1}$ from $\widehat{T}_h$
10: **end for**

Our high-level sequential transfer procedure, which combines PTUM with HMM learning, is presented in Algorithm 2. The approach keeps a running estimate of all MDP models $\{\widetilde{\mathcal{M}}^h_\theta\}_{\theta\in\Theta}$ together with a bound on their errors Δ_h. These estimates are used to solve each task via PTUM (line 4) and updated by plugging the resulting observations into the spectral method (lines 5-7). Then, the error bounds of the new estimates are computed as explained in Section 4.1 (line 8). Finally, the estimated task-transition matrix is used to predict which tasks are more likely to occur and the resulting set $\widetilde{\Theta}_{h+1}$ is used to run PTUM at the next step.

4.1. Error Bounds for the Estimated Models

We now derive finite-sample bounds on the estimation error of each MDP model. First, we need the following assumption, due to Anandkumar et al. (2012), to ensure that we can recover the HMM parameters from the samples.

Assumption 2. *The matrix O is full column-rank. Furthermore, the stationary distribution of the underlying Markov*

chain, $\omega \in \mathcal{P}(\Theta)$, is such that $\omega(\theta) > 0$ for all $\theta \in \Theta$.

As noted by Anandkumar et al. (2012), this assumption is milder than assuming minimal positive gaps between the different models (as was done, e.g., by Brunskill & Li (2013) and Liu et al. (2016)). We now present the main result of this section, which provides deviation inequalities between true and estimated models that hold uniformly over time.

Theorem 2. *Let $\{\widetilde{\mathcal{M}}_\theta^h\}_{\theta\in\Theta,h\geq 1}$ be the sequence of MDP models estimated by Algorithm 2, with $\widetilde{p}_{h,\theta}(s'|s,a)$ and $\widetilde{q}_{h,\theta}(u|s,a)$ the transition and reward distributions, $\widetilde{V}^*_{h,\theta}$ the optimal value functions, and $\widetilde{\sigma}^r_{h,\theta}(s,a)$, $\widetilde{\sigma}^p_{h,\theta}(s,a;\theta')$ the corresponding variances. There exist constants $\rho, \rho_r, \rho_p, \rho_{\sigma_r}, \rho_{\sigma_p}$ such that, for $\delta' \in (0,1)$, if*

$$h > \rho \log \frac{2h^2 SA(S+U)}{\delta'},$$

then, with probability at least $1-\delta'$, the following hold simultaneously for all $h \geq 1$, $s \in \mathcal{S}$, $a \in \mathcal{A}$, and $\theta, \theta' \in \Theta$:

$$|r_\theta(s,a) - \widetilde{r}_{h,\theta}(s,a)| \leq \rho_r \sqrt{\frac{\log c_{h,\delta'}}{h}},$$

$$|(p_\theta(s,a) - \widetilde{p}_{h,\theta}(s,a))^T \widetilde{V}^*_{h,\theta'}| \leq \rho_p \sqrt{\frac{\log c_{h,\delta'}}{h}},$$

$$|\sigma^r_\theta(s,a) - \widetilde{\sigma}^r_{h,\theta}(s,a)| \leq \rho_{\sigma_r} \sqrt{\frac{\log c_{h,\delta'}}{h}},$$

$$|\sigma^r_\theta(s,a;\theta') - \widetilde{\sigma}^p_{h,\theta}(s,a;\theta')| \leq \rho_{\sigma_p} \sqrt{\frac{\log c_{h,\delta'}}{h}},$$

where $c_{h,\delta'} := \pi^2 h^2 SA(S+U)/\delta'$.

To favor readability, we collapsed all terms of minor relevance to our purpose into the constants ρ. These are functions of the given family of tasks (through maximum/minimum eigenvalues of the covariance matrices introduced previously) and of the underlying Markov chain. Full expressions are reported in Appendix D. These bounds provide us with the desired deviations between the true and estimated models and can be used as input to the PTUM algorithm to learn each task in the sequential setup. It is important to note that, like other applications of spectral methods to RL problems (Azar et al., 2013a; Azizzadenesheli et al., 2016), the constants ρ are not known in practice and should be regarded as a parameter of the algorithm. Such a parameter can be interpreted as the confidence level in a standard confidence interval. Finally, regardless of constants, these deviations decay at the classic rate of $\mathcal{O}(1/\sqrt{h})$, so that the agent recovers the perfect models of all tasks in the asymptotic regime. This also implies that the agent needs to observe $\mathcal{O}(1/\epsilon^2)$ tasks before PTUM enters the transfer mode, which is theoretically tight in the worst-case (Feng et al., 2019) but quite conservative in practice.

4.2. Exploiting the Inter-Task Dynamics

We now show how the agent can exploit the task-transition matrix T to further reduce the sample complexity. Suppose that T and the identity of the current task θ^*_h are known. Then, through $T(\cdot|\theta^*_h)$, the agent has prior knowledge about the next task and it could, for instance, discard all models whose probability is low enough without even passing them to the PTUM algorithm. Using this insight, we can design an approach to pre-process the initial set of models by replacing T with its estimate $\widehat{T}$. Let $\widehat{T}_h$ be the estimated task-transition matrix and $\bar{\Theta}_h$ be the set of active models returned by Algorithm 1. Then, by design of the algorithm, $\bar{\Theta}_h$ contains the target task θ^*_h with sufficient probability and thus we can predict the probability that θ^*_{h+1} is equal to some θ as $\sum_{\theta'\in\bar{\Theta}_h} \widehat{T}_h(\theta,\theta')$. Our idea is to check whether this probability, plus some confidence value, is lower than a certain threshold η. In such a case, we discard θ from the initial set of models $\widetilde{\Theta}_{h+1}$ to be used at the next step. The following theorem ensures that, for a well-chosen threshold value, the target model is never discarded from the initial set at any time with high probability.

Theorem 3. *Let $\delta' \in (0,1)$ and $\delta \leq \frac{\delta'}{3m^2}$ be the confidence value for Algorithm 1. Suppose that, before each task h, a model θ is eliminated from the initial active set if:*

$$\sum_{\theta'\in\bar{\Theta}_h} \widehat{T}_h(\theta,\theta') + \delta k + \rho_T k \sqrt{\frac{\log(9kdm^2/\delta')}{h}} \leq \eta.$$

Then, for $\eta = \frac{\delta'}{3km^2}$, with probability at least $1-\delta'$, at any time the true model is never eliminated from the initial set.

As before, we collapsed minor terms into the constant ρ_T. When T is sparse (e.g., only a small number of successor tasks is possible), this technique could significantly reduce the sample complexity to identify a near-optimal policy as many models are discarded even before starting PTUM.

5. Related Works

In recent years, there has been a growing interest in understanding transfer in RL from a theoretical viewpoint. The work by Brunskill & Li (2013) and its follow-up by Liu et al. (2016) are perhaps the most related to ours. The authors consider a multi-task scenario in which the agent faces finitely many tasks drawn from a fixed distribution. Although their methods are designed for task identification, they do not actively seek information as our algorithm does and the resulting guarantees are therefore weaker. Azar et al. (2013a) present similar ideas in the context of multi-armed bandits. Like our work, they employ spectral methods to learn tasks sequentially, without however exploiting any temporal correlations. Dyagilev et al. (2008) propose a method for task identification based on sequential hypothesis testing. Their algorithm explores the environment by

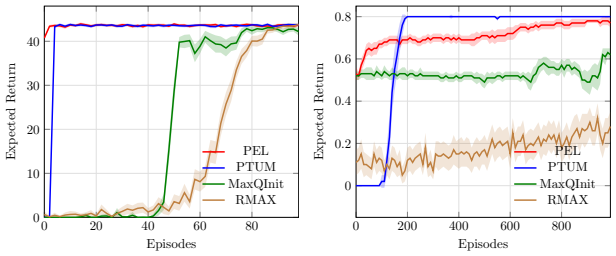

Figure 1. Policy transfer from known models. *(left)* Optimism gains information. *(right)* High-reward states are poorly informative.

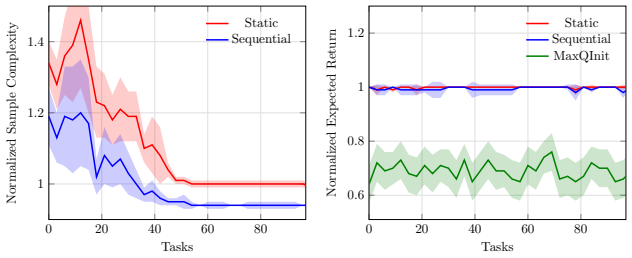

Figure 2. Sequential transfer experiment. *(left)* The normalized sample complexity. *(right)* Performance of the computed policies.

running optimistic policies and, like ours, aims at eliminating models from a confidence set. Mann & Choe (2012) and Abel et al. (2018) study how to to achieve jumpstart. The idea is to optimistically initialize the value function of the target task and then run any PAC algorithm. The resulting methods provably achieve positive transfer. Mahmud et al. (2013) cluster solved tasks in a few groups to ease policy transfer, which relates to our spectral learning of MDP models. Our setup is also related to other RL settings, including contextual MDPs (Hallak et al., 2015; Jiang et al., 2017; Modi et al., 2018; Sun et al., 2018), where the decision process depends on latent contexts, and hidden-parameter MDPs (Doshi velez & Konidaris, 2013; Killian et al., 2017), which follow a similar idea. Finally, we notice that the PAC-MDP framework has been widely employed to study RL algorithms (Strehl et al., 2009). Many of the above-mentioned works take inspiration from classic PAC methodologies, including the well-known E3 (Kearns & Singh, 2002), R-MAX (Brafman & Tennenholtz, 2002), and Delayed Q-learning (Strehl et al., 2006), and so does ours.

6. Experiments

The goal of our experiments is twofold. First, we analyze the performance of PTUM when the task models are known, focusing on the comparison between identification and jumpstart strategies. Then, we consider the sequential setting and show that our approach progressively transfers knowledge and exploits the temporal task correlations to improve over "static" transfer methods. Due to space constraints, here we present only the high-level setup of each experiment and discuss the results. We refer the reader to Appendix E for all the details and further results. All our plots report averages of 100 runs with 99% Student's t confidence intervals.

Identification vs Jumpstart For this experiment, we adopt a standard 12×12 grid-world divided into two parts by a vertical wall (i.e., the 4-rooms domain of Sutton et al. (1999) with only two rooms). The agent starts in the left room and must reach a goal state in the right one, with the two rooms connected by a door. We consider 12 different tasks, whose models are known to the agent, with different goal and door locations. We compare PTUM with three baselines: RMAX (Brafman & Tennenholtz, 2002), which does not perform any transfer, RMAX with MaxQInit (Abel et al., 2018), in which the Q-function is initialized optimistically to achieve a jumpstart, and PEL (Dyagilev et al., 2008), which performs model identification. Not to give advantage to PTUM, as the other algorithms run episodes online with no generative model, in our plots we count each sample taken by PTUM as one episode and assign it the minimum reward value. Once PTUM returns a policy, we report its online performance. The results in Figure 1*(left)* show that the two identification methods find an optimal policy in a very small number of episodes. The jumpstart of MaxQInit is delayed since the optimistic Q-function directs the agent towards the wall, but finding the door takes many samples. PEL, which is also optimistic, instantly identifies the solution since attempts to cross the wall in locations without a door immediately discriminate between tasks. This is in fact an example where the optimism principle leads to both rewards and information. To understand why this is not always the case, we run the same experiment by removing the wall and adding multiple goal locations with different reward values. Some goals have uniformly high values over all tasks (thus they provide small information) while others are more variable (thus with higher information). In Figure 1*(right)*, we see that both PEL and MaxQInit achieve a high jumpstart, as they immediately go towards one of the goals, but converge much more slowly than PTUM, which focuses on the informative ones.

Sequential Transfer We consider a variant of the object-world domain by Levine et al. (2011). Here, we have a 5×5 grid where each cell can contain one of 11 possible items with different values (or no items at all). There are 8 possible tasks, each of which randomizes the items with probabilities inversely proportional to their value. In order to show the benefits of exploiting temporal correlations, we design a sparse task-transition matrix in which each task has only a few possible successors. We run our sequential transfer method (Algorithm 2) and compare it with a variant that ignores the inter-task dynamics and only performs "static" transfer (i.e., only the estimated MDP models

are used). We note that model-learning in this variant reduces to the spectral method proposed by Azar et al. (2013a) for bandits. For the initial phase of the learning process, where the transfer condition is not satisfied, we solve each task by sampling state-action pairs uniformly as in Azar et al. (2013b). Figure 2*(left)* shows the normalized number of samples required by PTUM to solve each task starting from the first step in which the algorithm enters the transfer mode[4]. We can appreciate that the sample complexities of both variants decay as the number of tasks grows, meaning that the learned models become more and more accurate. As expected, the sequential version outperforms the static one by exploiting the temporal task correlations. It is important to note that the normalized sample complexity of our algorithm goes below 1 (i.e., below the oracle). This is a key point as the sample complexity of oracle PTUM is computed with the set of all models as input, while the sequential approach filters this set based on the estimated task-transition matrix. In Figure 2*(right)*, we confirm that both variants of our approach return near-optimal policies at all steps (all are ϵ-optimal for the chosen value of ϵ). We also report the performance of the policies computed by MaxQInit when the same maximum-sample budget as our approach is given (to be fair, we overestimated it as 5 to 10 times larger than what oracle PTUM needs). Although for MaxQInit we used the oracle optimistic initialization (with the true Q-functions), the algorithm is not able to obtain near-optimal performance using the given budget, while PTUM achieves it despite the estimated models.

7. Conclusion

We studied two important questions in lifelong RL. First, we addressed the problem of quickly identifying a near-optimal policy to be transferred among the solutions of previously-solved tasks. The proposed approach sheds light on how to design strategies that actively seek information to reduce identification time, and our sample complexity bounds confirm a significant positive transfer. Second, we showed that learning sequentially-related tasks is not significantly harder than the common case where they are statically sampled from a fixed distribution. By combining these two building blocks, we obtained an algorithm that sequentially learns the task structure, exploits temporal correlations, and quickly identifies the solution of each new task. We confirmed these properties in simple simulated domains, showing the benefits of identification over jumpstart strategies and of exploiting temporal task relations.

Our work opens up several interesting directions. Our ideas for task identification could be extended to the online setting, where a generative model is not available. Here, we would probably need a more principled trade-off between information and rewards, which can be modeled by different objectives (such as regret). Furthermore, it would be interesting to study the sequential transfer setting in large continuous MDPs. Here we could take inspiration from recent papers on meta RL that learn latent task representations (Humplik et al., 2019; Lan et al., 2019; Rakelly et al., 2019; Zintgraf et al., 2019) and meta-train policies that seek information for quickly identifying the task parameters.

[4]The sample complexity to solve each task is divided by the one of the oracle version of PTUM with known models

References

Abel, D., Jinnai, Y., Guo, S. Y., Konidaris, G., and Littman, M. Policy and value transfer in lifelong reinforcement learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 20–29, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Anandkumar, A., Hsu, D., and Kakade, S. M. A method of moments for mixture models and hidden markov models. In *Conference on Learning Theory*, pp. 33–1, 2012.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

Azar, M., Lazaric, A., and Brunskill, E. Sequential transfer in multi-armed bandit with finite set of models. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2220–2228. Curran Associates, Inc., 2013a.

Azar, M., Munos, R., and Kappen, H. J. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine Learning*, 91(3): 325–349, Jun 2013b.

Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. Reinforcement learning of pomdps using spectral methods. *arXiv preprint arXiv:1602.07764*, 2016.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4055–4065. Curran Associates, Inc., 2017.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. Transfer in deep reinforcement learning using successor

features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510, 2018.

Boucheron, S., Lugosi, G., and Bousquet, O. Concentration inequalities. In *Summer School on Machine Learning*, pp. 208–240. Springer, 2003.

Brafman, R. I. and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002.

Brunskill, E. and Li, L. Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821*, 2013.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. An environment model for nonstationary reinforcement learning. In *Advances in neural information processing systems*, pp. 987–993, 2000a.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pp. 264–287. Springer, 2000b.

Doshi velez, F. and Konidaris, G. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. *IJCAI : proceedings of the conference*, 2016, 08 2013.

Dyagilev, K., Mannor, S., and Shimkin, N. Efficient reinforcement learning in parameterized models: Discrete parameter case. In *European Workshop on Reinforcement Learning*, pp. 41–54. Springer, 2008.

Feng, F., Yin, W., and Yang, L. F. Does knowledge transfer always help to learn a better policy? *arXiv preprint arXiv:1912.02986*, 2019.

Fernández, F. and Veloso, M. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 720–727, 2006.

Guo, Z. D., Doroudi, S., and Brunskill, E. A pac rl algorithm for episodic pomdps. In *Artificial Intelligence and Statistics*, pp. 510–518, 2016.

Hadoux, E., Beynier, A., and Weng, P. Solving hidden-semi-markov-mode markov decision problems. In *International Conference on Scalable Uncertainty Management*, pp. 176–189. Springer, 2014.

Hallak, A., Di Castro, D., and Mannor, S. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

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.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

Jian, Q., Fruit, R., Pirotta, M., and Lazaric, A. Exploration bonus for regret minimization in discrete and continuous average reward mdps. In *Advances in Neural Information Processing Systems*, pp. 4890–4899, 2019.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. Contextual decision processes with low bellman rank are pac-learnable. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1704–1713. JMLR. org, 2017.

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

Killian, T. W., Daulton, S., Konidaris, G., and Doshi-Velez, F. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 6250–6261, 2017.

Konidaris, G., Scheidwasser, I., and Barto, A. Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research*, 13(May):1333–1371, 2012.

Lan, L., Li, Z., Guan, X., and Wang, P. Meta reinforcement learning with task embedding and shared policy. *arXiv preprint arXiv:1905.06527*, 2019.

Lazaric, A. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pp. 143–173. Springer, 2012.

Lazaric, A., Restelli, M., and Bonarini, A. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 544–551. ACM, 2008.

Levine, S., Popovic, Z., and Koltun, V. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2011.

Liu, Y. and Stone, P. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the 21st national conference on Artificial intelligence-Volume 1*, pp. 415–420, 2006.

Liu, Y., Guo, Z., and Brunskill, E. Pac continuous state online multitask reinforcement learning with identification.

In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 438–446, 2016.

Mahmud, M., Hawasly, M., Rosman, B., and Ramamoorthy, S. Clustering markov decision processes for continual transfer. *arXiv preprint arXiv:1311.3959*, 2013.

Mann, T. A. and Choe, Y. Directed exploration in reinforcement learning with transferred knowledge. 2012.

Maurer, A. and Pontil, M. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

Meng, L. and Zheng, B. The optimal perturbation bounds of the moore–penrose inverse under the frobenius norm. *Linear algebra and its applications*, 432(4):956–963, 2010.

Modi, A., Jiang, N., Singh, S., and Tewari, A. Markov decision processes with continuous side information. In *Algorithmic Learning Theory*, pp. 597–618, 2018.

Pukelsheim, F. *Optimal design of experiments*. SIAM, 2006.

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

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, 2019.

Rosman, B., Hawasly, M., and Ramamoorthy, S. Bayesian policy reuse. *Machine Learning*, 104(1):99–127, 2016.

Sidford, A., Wang, M., Wu, X., Yang, L., and Ye, Y. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems*, pp. 5186–5196, 2018.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888, 2006.

Strehl, A. L., Li, L., and Littman, M. L. Reinforcement learning in finite mdps: Pac analysis. *Journal of Machine Learning Research*, 10(Nov):2413–2444, 2009.

Sun, S. A survey of multi-view machine learning. *Neural computing and applications*, 23(7-8):2031–2038, 2013.

Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., and Langford, J. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. *arXiv preprint arXiv:1811.08540*, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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

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

Taylor, M. E. and Stone, P. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Taylor, M. E., Jong, N. K., and Stone, P. Transferring instances for model-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 488–505. Springer, 2008.

Tirinzoni, A., Rodriguez Sanchez, R., and Restelli, M. Transfer of value functions via variational methods. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6179–6189. Curran Associates, Inc., 2018a.

Tirinzoni, A., Sessa, A., Pirotta, M., and Restelli, M. Importance weighted transfer of samples in reinforcement learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4936–4945, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR.

Tirinzoni, A., Salvini, M., and Restelli, M. Transfer of samples in policy search via multiple importance sampling. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6264–6274, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Tirinzoni, A., Lazaric, A., and Restelli, M. A novel confidence-based algorithm for structured bandits. In Chiappa, S. and Calandra, R. (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3175–3185, Online, 26–28 Aug 2020. PMLR.

Yang, J., Petersen, B., Zha, H., and Faissol, D. Single episode policy transfer in reinforcement learning. *arXiv preprint arXiv:1910.07719*, 2019.

Yin, H. and Pan, S. J. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Zanette, A., Kochenderfer, M. J., and Brunskill, E. Almost horizon-free structure-aware best policy identification with a generative model. In *Advances in Neural Information Processing Systems*, pp. 5625–5634, 2019.

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. *arXiv preprint arXiv:1910.08348*, 2019.