Learning to Undo: Transfer Reinforcement Learning under Linear State Space Transformations

Mridul Mahajan[1], **Aldo Pacchiano**[1,2†], **Xuezhou Zhang**[1†]
{mridulm,pacchian,xuezhouz}@bu.edu

[1]**Boston University**
[2]**Broad Institute of MIT and Harvard**

† Equal senior authorship

Abstract

Transfer learning in reinforcement learning (RL) has shown strong empirical success. In this work, we take a more principled perspective by studying when and how transferring knowledge between MDPs can be provably beneficial. Specifically, we consider the case where there exists a linear undo map between two MDPs (a source and a target), such that applying this map to the target's state space recovers the source exactly. We propose an algorithm that learns this map via linear regression on state feature statistics gathered from both MDPs, and then uses it to obtain the target policy in a zero-shot manner from the source policy. Theoretically, we show that for linear MDPs, our approach has strictly better sample complexity than learning from scratch. Empirically, we demonstrate that these benefits extend beyond the linear setting: on challenging continuous control tasks, our method achieves significantly improved sample efficiency. Overall, our results highlight how shared structure between tasks can be leveraged to make learning more efficient.

1 Introduction

Reinforcement learning (RL) has seen rapid progress in recent years and has achieved strong performance in complex tasks, such as video games, locomotion, manipulation, and navigation (Mnih et al., 2015; Lee et al., 2019; Zhu et al., 2019). Despite this, learning high-quality policies from scratch typically requires millions of environment interactions. This inefficiency limits the applicability of RL in real-world domains where data collection is costly or time-consuming (Dulac-Arnold et al., 2019).

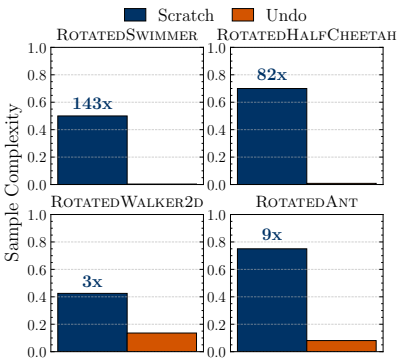

Figure 1: Sample complexity (in million) to reach 95% of optimal performance. Numbers on blue bars quantify how much worse learning from scratch is.

Transfer learning aims to overcome this challenge by leveraging knowledge learned from source tasks to accelerate learning on related target tasks (Taylor & Stone, 2009). It has the potential to drastically improve sample efficiency by effectively reusing prior experience. While transfer learning has seen strong empirical success in RL (Zhu et al., 2020), we study the problem from a more principled lens, where we explicitly model the structured similarity between tasks and exploit it for transfer.

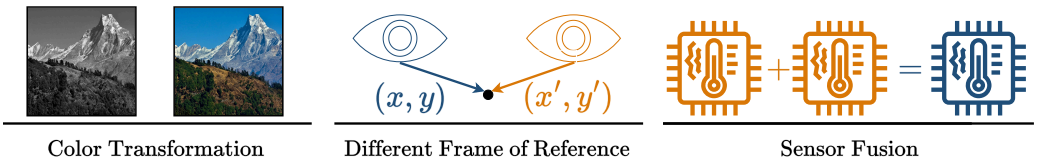

Figure 2: Linear transformations are grounded in real-world applications.

An interesting setting is that of linear transformations: the source and target tasks differ due to transformations in the state space that are linear. For instance, real-world transformations such as color transformation from RGB to grayscale, change in frame of reference, and sensor fusion – all are linear (see Fig. 2). Linear transformation provides a rich and practical setting to design principled transfer learning methods that explicitly account for these transformations.

In this work, we propose a novel approach to address this setting by learning an *undo map* that transforms the state space in the target back to the source. Instead of learning a policy from scratch, our method recovers a function that *undoes* the transformation applied to the target states. This allows us to reuse the source policies with minimal additional samples from the target task.

We propose an algorithm that exploits the fact that the action space is unchanged to learn this undo map via linear regression on state feature statistics gathered from both MDPs. It subsequently uses the undo map to obtain the target policy in a zero-shot manner by composing the source policy with the learned undo map. Theoretically, we show that for linear MDPs, our approach has strictly better sample complexity than learning from scratch.

To evaluate our method, we use challenging continuous control tasks, and construct target tasks by changing the frame of reference of their observations, as well as a setting inspired by sensor fusion. Our results show that learning the undo map consistently outperforms learning the target policy from scratch in terms of sample efficiency, where it often achieves near-optimal performance with an order of magnitude fewer samples (see Fig. 1). These results highlight the practical utility of our approach for transfer learning in RL under linear state-space transformations.

To summarize, our work makes the following key contributions:

I. We propose a novel transfer learning method that performs linear regression on state feature statistics to *undo* linear state space transformations in the target MDP (Section 3).

II. Theoretically, we prove that for linear MDPs, our approach achieves strictly better sample complexity than learning from scratch (Section 4).

III. We demonstrate the strong empirical performance of our method by conducting experiments on challenging continuous control tasks with linear state space transformations (Section 5).

2 Problem Setup

Tasks. We model each task as a finite-horizon episodic Markov Decision Process (MDP), defined by the 7-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, r, H, d_0, \phi)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ defines the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, H is the fixed horizon length, $d_0 \in \mathcal{P}(\mathcal{S})$ is the initial state distribution, and $\phi : \mathcal{S} \to \mathbb{R}^d$ is a fixed feature map. We assume that $d \leq H$. The agent interacts with the task using a non-stationary policy $\pi_h : \mathbb{R}^d \to \mathcal{P}(\mathcal{A})$, and for $0 \leq h < H$, chooses an action $a_h \sim \pi_h(\cdot \mid \phi(s_h))$. The value of the policy π is defined as the expected return $V^\pi \triangleq \mathbb{E}_{s_0 \sim d_0, \pi, \mathbb{P}} \left[\sum_{h=0}^{H-1} r(s_h, a_h) \right]$, and the agent's goal is to learn a policy $\pi^\star$ that maximizes this quantity, i.e., $V^* = \max_\pi V^\pi$. The state-action value function $Q^\pi(s, a)$ denotes the expected return when taking action a in state s, and following policy π thereafter.

Transfer Learning. The transfer learning setting follows the protocol in which the agent first interacts with a source task $\mathcal{M}_S = (\mathcal{S}^S, \mathcal{A}^S, \mathbb{P}^S, r^S, H, d_0^S, \phi_S)$ to learn a policy π_S, and then leverages the learned policy to speed up learning in the target task $\mathcal{M}_T = (\mathcal{S}^T, \mathcal{A}^T, \mathbb{P}^T, r^T, H, d_0^T, \phi_T)$. We assume that the agent has complete access to the source MDP $\mathcal{M}_S$, either by knowing its dynamics and reward function exactly or by being allowed to interact with it arbitrarily without incurring any sample complexity cost. In contrast, interactions with the target MDP $\mathcal{M}_T$ are limited and costly. $\mathcal{M}_S$ and $\mathcal{M}_T$ have some shared structure, which we describe below.

Shared Structure and Undo Map. We assume that both tasks are exactly the same except that the agent observes states through task-specific feature maps: $\phi_S : \mathcal{S} \to \mathbb{R}^{d_S}$ in the source and $\phi_T : \mathcal{S} \to \mathbb{R}^{d_T}$ in the target. We assume these feature maps are related by an unknown linear transformation (the *undo* map) $U_\star : \mathbb{R}^{d_T} \to \mathbb{R}^{d_S}$, such that for all $s \in \mathcal{S}$, $\phi_S(s) = U_\star \phi_T(s)$. This relationship induces a correspondence between policies: for any policy $\pi_S : \mathbb{R}^d \to \mathcal{P}(\mathcal{A})$ defined in the source feature space, we define the corresponding policy in the target task as $\pi_T\big(\phi_T(s)\big) = \pi_S\big(U_\star \phi_T(s)\big)$. This allows policies to generalize across tasks despite differing state representations.

Objective. Given complete access to $\mathcal{M}_S$, our objective is to learn a near-optimal policy for the target task using as few samples from the target as possible.

3 Learning the Undo Map via Regression

In this section, we present a learning algorithm that estimates the undo map from samples. Our method leverages the fact that the action spaces in the source and target MDPs are exactly the same. At a high level, the algorithm computes state feature statistics in the source and the target that are related via the undo map, and uses this relation for transfer learning.

Definition 1 (Expected State Feature Sum)**.** *Given an action sequence $a_{0:h-1}$, define $\psi_h(a_{0:h-1})$ as the h-step truncated sum of state features computed from a sample trajectory:*

$$\psi_h(a_{0:h-1}) \triangleq \sum_{t=0}^{h-1} \phi(s_t), \tag{1}$$

where $(s_0, s_1, \ldots, s_{h-1})$ is the state sequence obtained by executing actions $a_0, \ldots, a_{h-1}$. The expected state feature sum for h steps, denoted by $\bar{\psi}_h(a_{0:h-1})$, is defined as:

$$\bar{\psi}_h(a_{0:h-1}) \triangleq \mathbb{E}\left[\psi_h(a_{0:h-1})\right], \tag{2}$$

where the expectation is over the dynamics of the MDP.

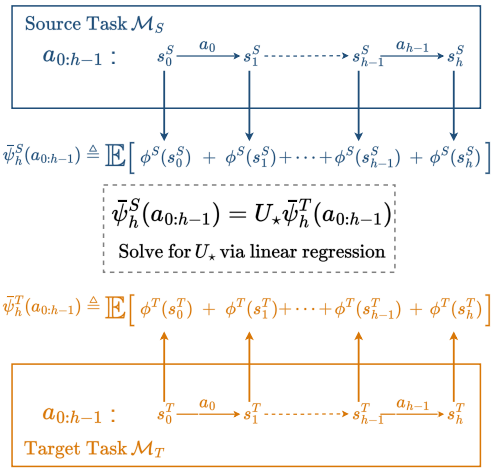

Figure 3: Overview of our method. We recover the map to *undo* the state space transformation by linear regression on state feature statistics.

Note that the expected state feature sum is defined with respect to an *open-loop* sequence of actions. Consequently, since the source and target MDPs differ only through a linear transformation of the state space, the expected state feature sums in the source and the target (denoted by $\bar{\psi}_h^S(a_{0:h-1})$ and $\bar{\psi}_h^T(a_{0:h-1})$, respectively) are related via the undo map $U_\star$ as follows:

$$\bar{\psi}_h^S(a_{0:h-1}) = U_\star \bar{\psi}_h^T(a_{0:h-1}),\ \forall h \in [H]. \tag{3}$$

As samples from the source MDP $\mathcal{M}_S$ are free, we can compute $\bar{\psi}_h^S(a_{0:h-1})$ exactly, while $\bar{\psi}_h^T(a_{0:h-1})$ can be empirically estimated using samples from the target MDP $\mathcal{M}_T$.

Algorithm 1 Estimate Undo Map via Ridge Regression

Require: Source MDP $\mathcal{M}_S$, target MDP $\mathcal{M}_T$, action sequence $a_{0:H-1}$, number of trajectories n, regularization parameter λ

1: Exactly compute the expected feature sum $\bar{\psi}_h^S(a_{0:h-1})$ in $\mathcal{M}_S$, $\forall h \in [H]$.
2: Roll out n trajectories in $\mathcal{M}_T$ using action sequence $a_{0:H-1}$.
3: **for** $h \in [H]$ **do**
4: For each trajectory i, compute feature sum $\psi_h^{T,(i)}(a_{0:h-1})$.
5: Compute empirical average:

$$\widehat{\psi}_h^T(a_{0:h-1}) = \frac{1}{n}\sum_{i=1}^{n} \psi_h^{T,(i)}(a_{0:h-1}).$$

6: **end for**
7: Solve for $\widehat{U} \in \mathbb{R}^{d_S \times d_T}$ (which estimates $U_\star$) via ridge regression:

$$\widehat{U} = \arg\min_{U \in \mathbb{R}^{d_S \times d_T}} \sum_{h=1}^{H} \left\| \bar{\psi}_h^S(a_{0:h-1}) - U\widehat{\psi}_h^T(a_{0:h-1}) \right\|_2^2 + \lambda \|U\|_F^2$$

8: **return** $\widehat{U}$

We now describe how to learn the undo map from target samples.

Learning via Ridge Regression. The relations between expected state feature sums in the source and the target are equivalent to a system of linear equations (one for each time horizon $h \in [H]$), which we use to estimate $U_\star$ via ridge regression. Since $U_\star \in \mathbb{R}^{d_S \times d_T}$, estimating it amounts to solving d_S separate linear regression problems, each with d_T unknowns. Therefore, we need at least d_T linearly independent equations to ensure identifiability. Since $d_T < H$ (by assumption), we use the expected feature sum relation for each $h \in [H]$ to construct H such equations. Given that $\bar{\psi}_h^T$ is estimated empirically, we adopt ridge regression to mitigate the impact of estimation noise. Specifically, let $\widehat{\psi}_h^T(a_{0:h-1})$ denote the empirical estimate of the expected target feature sum obtained from n trajectories. Then the undo map is estimated by solving:

$$\widehat{U} = \arg\min_{U \in \mathbb{R}^{d_S \times d_T}} \sum_{h=1}^{H} \left\| \bar{\psi}_h^S(a_{0:h-1}) - U\widehat{\psi}_h^T(a_{0:h-1}) \right\|_2^2 + \lambda \|U\|_F^2, \tag{4}$$

where $\lambda > 0$ is a regularization parameter and $\|\cdot\|_F$ denotes the Frobenius norm.

The estimate $\hat{u}$ allows us to *lift* the optimal policy for the source task $\pi_S^\star$ to an optimal policy for the target task: $\pi_T^\star(\cdot) = \pi_S^\star(\widehat{U}(\cdot))$.

The pseudocode for the proposed algorithm to learn the undo map is given in Algorithm 1. We would like to comment that our method is generally applicable even when the undo map is non-linear.

4 Theoretical Analysis

We model both the source and target tasks as linear MDPs, which we define below.

Definition 2 (Linear MDP (Jin et al., 2020))**.** *An MDP is linear if the transition probabilities exhibit a low-rank decomposition: $\mathbb{P} = \Phi \times \Psi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times|\mathcal{S}|}$, where $\Phi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times d}$ and $\Psi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|\times d}$, and the matrix Φ is known to the learner. Each row of Φ is denoted as $\phi(s,a) \in \mathbb{R}^d$, which represents the state-action features.*

We now list the assumptions used in our analysis. The first is a standard assumption in linear MDPs.

Assumption 1. *The reward function R is linear in the feature vector $\phi(s,a)$. Specifically, for each state-action pair (s,a), the reward is given by $r(s,a) = \phi(s,a)^\top \theta_R$, where $\theta_R \in \mathbb{R}^d$.*

Algorithm 2 Estimate $U_\star$ via Least Squares for Linear MDPs

Require: Target MDP $\mathcal{M}_T$, expected feature sum oracle $\mathcal{O}$, action sequence $a_{0:H-1}$, number of trajectories n

1: Query $\mathcal{O}$ to obtain the expected feature sum $y_h = \bar{\psi}_h^S(a_{0:h-1})$ in $\mathcal{M}_S$, $\forall h \in [H]$.
2: Roll out n trajectories in $\mathcal{M}_T$ using action sequence $a_{0:H-1}$.
3: **for** $h \in [H]$ **do**
4: For each trajectory i, compute feature sum $x_{i,h} = \psi_h^{T,(i)}(a_{0:h-1})$.
5: **end for**
6: Solve for $\widehat{U} \in \mathbb{R}^{d_S \times d_T}$ (which estimates $U_\star$) via ordinary least squares regression:

$$\widehat{U}[i] = \widehat{\Lambda_n}^{-1} \sum_{h=1}^{H} \sum_{j=1}^{n} x_{j,h} y_h[i],$$

where $\widehat{\Lambda_n} = \sum_{h=1}^{H} \sum_{j=1}^{n} x_{j,h} x_{j,h}^T$.
7: **return** U

This linear reward assumption implies that the action-value function $Q^\pi(s,a)$ for any policy π is also linear in $\phi(s,a)$. We further assume that $|\phi(s,a)|_2 \leq 1$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, and that rewards are bounded within the range $[-1,1]$.

In this setup, given an action sequence $a_{0:h-1}$, we define $\psi_h(a_{0:h-1})$ to be the average of the state-action features over the first h steps of a sample trajectory.:

$$\psi_h(a_{0:h-1}) \triangleq \frac{1}{h} \sum_{t=0}^{h-1} \phi(s_t, a_t). \tag{5}$$

The source MDP $\mathcal{M}_S$ and the target MDP $\mathcal{M}_T$ are exactly the same except for the state-action features (Φ_S in source and Φ_T in the target), which are related via a linear undo map $U_\star$: $\Phi_S = \Phi_T U_\star^T$. We assume that the ℓ_2 norm of each row of $U_\star$ is upper bounded by m.

The learner has access to the parameter vector $\theta_S^\star$ for the state-action values of the optimal source policy, and an oracle $\mathcal{O}$ for the state-action feature sum, as defined below:

Assumption 2 (Oracle for state-action feature sum in $\mathcal{M}_S$)**.** *We denote by $\mathcal{O}$ an oracle that when queried with an action sequence $a_{0:h-1}$ and time $h \in [H]$, outputs $\bar{\psi}_h^S(a_{0:h-1}) \triangleq \mathbb{E}\left[\psi_h^S(a_{0:h-1})\right]$.*

4.1 Transfer Learning Algorithm

On a high level, our learning algorithm estimates $U_\star$ by utilizing the following relation between the expected state-action feature sums in the source and the target:

$$\bar{\psi}_h^S(a_{0:h-1}) = U_\star \bar{\psi}_h^T(a_{0:h-1}). \tag{6}$$

More concretely, it first queries $\mathcal{O}$ to obtain $\bar{\psi}_h^S(a_{0:h-1})$ in $\mathcal{M}_S$, for all $h \in [H]$. Next, it rolls out n trajectories in the target using the open-loop action sequence $a_{0:H-1}$. It then views the truncated sum of state-action features $\psi_h^{T,(i)}(a_{0:h-1})$ (computed on the i-th sample trajectory) as noisy features for $\bar{\psi}_h^S(a_{0:h-1})$. Subsequently, it estimates $U_\star$ with $\widehat{U}$ via ordinary least squares regression.

The pseudocode for this algorithm is given in Algorithm 2.

4.2 Analysis

We will now show that in the setup described above, learning the undo map via linear regression on state-action feature statistics is provably better than learning from scratch.

From Eq. 6, the following relation holds:

$$Q_T^\star = \Phi_T U_\star^T \theta_S^\star. \tag{7}$$

Thus, a natural way to evaluate the output $\widehat{U}$ of Algorithm 2, is to bound the performance difference with respect to the optimal policy in the target, if we act greedily with respect to $\Phi_T \widehat{U}^T \theta_S^\star$. Let $\hat{\pi}$ denote this policy. We show that our algorithm has the following performance difference guarantee.

Theorem 1 (Performance Difference Bound for Algorithm 2)**.** *Define the covariance matrix*

$$\Sigma \triangleq \frac{1}{H} \sum_{h=1}^{H} \psi_h^T(a_{0:h-1}) \psi_h^T(a_{0:h-1})^T,$$

and let $\lambda_{\min}$ *denote* $\lambda_{\min}(\Sigma)$.

For all $n \geq \frac{6}{H\lambda_{\min}}(\log \frac{3d_T d_S}{\delta})$*, with probability at least* $1-\delta$*, the performance difference satisfies*

$$\|V_T^\star - V_T^{\hat{\pi}}\|_\infty \leq 2\sqrt{\frac{H}{n\lambda_{\min}}} \sqrt{d_T + 2\log\left(\frac{3d_S}{\delta}\right) + 2\sqrt{d_T \log\left(\frac{3d_S}{\delta}\right)}} + o\left(\sqrt{\frac{H}{n\lambda_{\min}}}\right).$$

The proof is provided in Appendix B.

The theorem shows that if the action sequence $a_{0:H-1}$ is *good*, i.e., $\lambda_{\min}(\Sigma)$ is sufficiently large (on the order of $1/d_T$), then Algorithm 2 achieves a sample complexity upper bound that scales as $\widetilde{O}(d_T\sqrt{H})$. This improves upon the minimax lower bound for linear MDPs, which scales as $\widetilde{O}(d_T H^{3/2})$ (He et al., 2023).

Remark. Our proof relies solely on the property that the optimal state-action value function is linearly realizable. Therefore, our performance difference guarantee extends to a more general setting where the only assumption is that the optimal state-action value function lies within the span of a known feature map. In this setting, the lower bound scales exponentially with the feature dimension d_T or the horizon H (Weisz et al., 2021), whereas our transfer learning method still enjoys $\widetilde{O}(d_T\sqrt{H})$ dependence.

5 Experimental Evaluation

In this section, we empirically evaluate our transfer learning method to address the following research questions: (i) Is it better to learn the undo map $U_\star$ than to learn the target policy from scratch? (ii) How critical is the choice of the *open-loop* action sequence $a_{0:H-1}$?

We begin by explaining the rationale for task selection, describing both the source and target tasks. Next, we provide an overview of the training process for the undo map, followed by the results.

5.1 Environments

We evaluate our method on continuous control environments that are challenging to learn from scratch: SWIMMER, HALFCHEETAH, WALKER2D, and ANT (Towers et al., 2024). To test the transfer, we construct corresponding target tasks by rotating the global frame of reference by 45° around relevant axes. This results in the tasks: ROTATEDSWIMMER, ROTATEDHALFCHEETAH, ROTATEDWALKER2D, and ROTATEDANT. Additionally, we design target tasks inspired by sensor fusion scenarios, resulting in the following tasks: FUSIONSWIMMER, FUSIONHALFCHEETAH, FUSIONWALKER2D, and FUSIONANT. These transformations induce structured transformations in the observation space without altering the dynamics of the original environment. Appendix C describes the target tasks in detail.

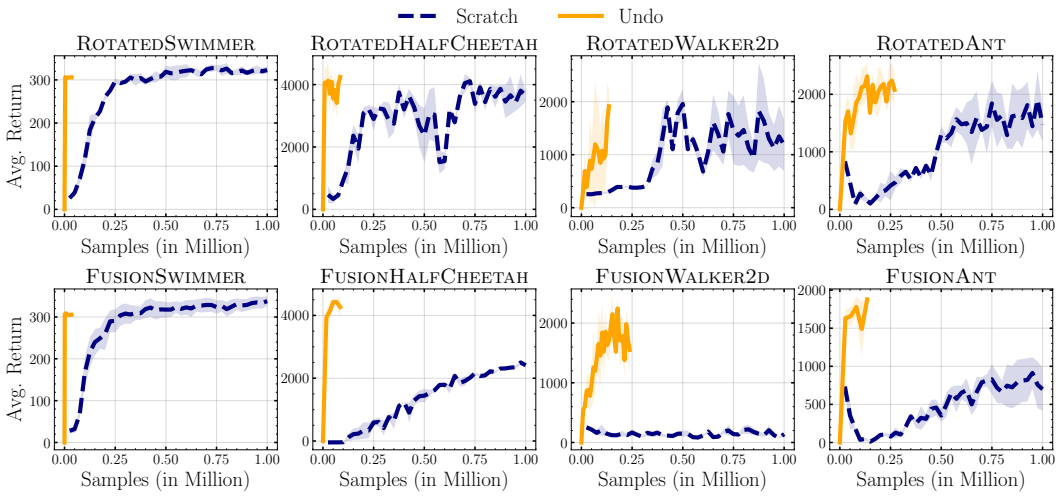

Figure 4: Comparison of avg. reward with varying budget of target samples. Transfer learning via the undo map consistently achieves significantly better sample efficiency than learning from scratch.

5.2 Training Procedure

We use the `rl-baselines3-zoo` framework (Raffin, 2020) to train both the source policies and the target policies used in the learning-from-scratch baseline, with PPO (Schulman et al., 2017) as the learning algorithm. For each environment, we adopt the tuned hyperparameters provided in the framework, training the policies for 1 million steps each. To estimate the expected state feature sums in the target MDP $\mathcal{M}_T$, rather than executing full episodes, we truncate rollouts to d_T steps. This yields a sufficient number of equations to estimate the undo map. We collect samples from approximately 5,000 rollouts, each of length d_T, where d_T is the dimensionality of the target observation space. For the source statistics, a similar number of samples suffices.

The action sequence $a_{0:H-1}$ used to compute the state feature statistics is derived from the source policy π_S. More concretely, we perform a rollout of π_S in the source MDP $\mathcal{M}_S$ and record the sequence of actions taken by the policy. We then use this recorded sequence as $a_{0:H-1}$.

5.3 Results

In this section, we first compare our method (listed as *Undo*) against learning the target policy from scratch (listed as *Scratch*). Next, we evaluate the criticality of the choice of the action sequence.

Comparison with Learning from Scratch. Fig. 4 reports the avg. reward, averaged over three random seeds, for each method under varying target sample budgets. Our method significantly improves sample efficiency across all tasks versus learning from scratch. Notably, *Undo* reaches near-optimal performance within just 0.1 million samples, whereas *Scratch* requires substantially more. Furthermore, the sensor fusion inspired target tasks are challenging to learn from scratch, which is likely due to the increased dimensionality of the state features. Nonetheless, *Undo* still achieves near-optimal performance with very few samples from the target.

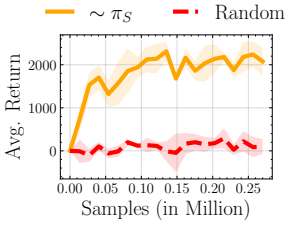

Figure 5: Sensitivity to action sequence on RotatedAnt. The choice of action sequence critically affects the performance of our method.

Choice of Action Sequence. Fig. 5 illustrates the sensitivity of our method to the choice of action sequence $a_{0:H-1}$. *Random* denotes results when the H actions are chosen uniformly at random, and $\sim \pi_S$ represents an action sequence collected by rolling out the source policy in $\mathcal{M}_S$. These results emphasize the critical role of selecting a good action sequence.

6 Related Work

Transfer Learning. Empirical methods for transfer learning in RL often lack provable guarantees or rely on application-specific assumptions that do not generalize. For instance, while (Sun et al., 2022) assume shared latent dynamics across observation spaces and that target observations can be transformed into the source domain, they do not learn this mapping explicitly and offer no theoretical guarantees on transfer performance. Similarly, (Watahiki et al., 2024) assume a shared latent MDP structure between tasks but their method is not theoretically grounded. Other works, such as (Agarwal et al., 2022), study transfer through reusing prior policies across design iterations, which is orthogonal to our setting. Methods like (Chen et al., 2024; Yi et al., 2023) rely on very specific visual or object-level assumptions, which limits their general applicability. The notion of an *undo* map to reverse state space transformations in reinforcement learning was first introduced in (Gupta et al., 2022), where the problem is approached through a distributional lens by aligning trajectories across the source and target tasks. However, the method does not scale beyond toy tabular environments.

Representational Transfer in Low-Rank MDPs. Several works have studied representational transfer under low-rank or linear MDP assumptions. For instance, (Agarwal et al., 2023; Cheng et al., 2022) use reward-free exploration in the source task to learn a good representation for the target. (Sam et al., 2024) extend this to a more general low-rank setting. (Lu et al., 2021) learn a linear representation using least squares for multitask linear MDPs, and (Ishfaq et al., 2024) consider the offline multitask case.

In contrast, our work approaches transfer RL through a more principled lens by making concrete assumptions about the structural relationship between the source and target tasks. We propose an algorithm that explicitly learns a mapping between their observation spaces by framing the problem as supervised learning, and therefore eliminates the need for RL in the target task. Under certain assumptions, this leads to provable improvements in sample complexity over learning from scratch. Moreover, our method is practical and demonstrates strong empirical performance. Essentially, our work takes a step toward more principled and broadly applicable approaches to transfer in RL.

7 Concluding Discussion

In this paper, we introduced a principled approach to transfer learning in reinforcement learning, where the source and target tasks are related by linear transformations of the state space. By explicitly learning an undo map, our method achieves significant gains in sample efficiency compared to learning target policies from scratch, as demonstrated across multiple challenging continuous control environments. Theoretically, we showed that under certain assumptions in linear MDPs, our approach achieves strictly better sample complexity than learning from scratch. A current limitation of our analysis is the reliance on a *good* action sequence – one that satisfies specific coverage properties required to estimate the undo map accurately. A promising direction for future work is to design principled algorithms that can discover such sequences. Extending the theoretical results beyond the linear MDP setting is another interesting direction. Finally, while our current evaluation focuses on linear transformations, our regression-based algorithm for learning the undo map is generally applicable, and it would be interesting to explore its ability to recover non-linear transformations.

References

Alekh Agarwal, Yuda Song, Wen Sun, Kaiwen Wang, Mengdi Wang, and Xuezhou Zhang. Provable Benefits of Representational Transfer in Reinforcement Learning. *COLT*, 2023.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Reincarnating Reinforcement Learning: Reusing Prior Computation to Accelerate Progress. *NeurIPS*, 2022.

Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, Chenfeng Xu, Quan Vuong, and Ken Goldberg. Mirage: Cross-Embodiment Zero-Shot Policy Transfer with Cross-Painting. *RSS*, 2024.

Yuan Cheng, Songtao Feng, Jing Yang, Hong Zhang, and Yingbin Liang. Provable Benefit of Multitask Representation Learning in Reinforcement Learning. *NeurIPS*, 2022.

Gabriel Dulac-Arnold, Daniel J. Mankowitz, and Todd Hester. Challenges of Real-World Reinforcement Learning. *CoRR*, abs/1904.12901, 2019.

Abhi Gupta, Ted Moskovitz, David Alvarez-Melis, and Aldo Pacchiano. Transfer RL via the Undo Maps Formalism. *CoRR*, abs/2211.14469, 2022.

Jiafan He, Heyang Zhao, Dongruo Zhou, and Quanquan Gu. Nearly Minimax Optimal Reinforcement Learning for Linear Markov Decision Processes. *ICML*, 2023.

Daniel J. Hsu, Sham M. Kakade, and Tong Zhang. Random Design Analysis of Ridge Regression. *Found. Comput. Math.*, 14(3):569–600, 2014.

Haque Ishfaq, Thanh Nguyen-Tang, Songtao Feng, Raman Arora, Mengdi Wang, Ming Yin, and Doina Precup. Offline Multitask Representation Learning for Reinforcement Learning. *NeurIPS*, 2024.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably Efficient Reinforcement Learning with Linear Function Approximation. *COLT*, 2020.

Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Sci. Robotics*, 4(26), 2019.

Rui Lu, Gao Huang, and Simon S Du. On the Power of Multitask Representation Learning in Linear MDP. *CoRR*, abs/2106.08053, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Antonin Raffin. RL Baselines3 Zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

Tyler Sam, Yudong Chen, and Christina Lee Yu. The Limits of Transfer Reinforcement Learning with Latent Low-rank Structure. *NeurIPS*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.

Yanchao Sun, Ruijie Zheng, Xiyao Wang, Andrew Cohen, and Furong Huang. Transfer RL across Observation Feature Spaces via Model-Based Regularization. *ICLR*, 2022.

Matthew E. Taylor and Peter Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *JMLR*, 2009.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *CoRR*, abs/2407.17032, 2024.

Hayato Watahiki, Ryo Iwase, Ryosuke Unno, and Yoshimasa Tsuruoka. Cross-Domain Policy Transfer by Representation Alignment via Multi-Domain Behavioral Cloning. *CoLLAs*, 2024.

Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential Lower Bounds for Planning in MDPs With Linearly-Realizable Optimal Action-Value Functions. *ALT*, 2021.

Qi Yi, Rui Zhang, Shaohui Peng, Jiaming Guo, Yunkai Gao, Kaizhao Yuan, Ruizhi Chen, Siming Lan, Xing Hu, Zidong Du, Xishan Zhang, Qi Guo, and Yunji Chen. Online Prototype Alignment for Few-shot Policy Transfer. *ICML*, 2023.

Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost. *ICRA*, 2019.

Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou. Transfer Learning in Deep Reinforcement Learning: A Survey. *IEEE TPAMI*, 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Table of Contents

In this section, we provide an outline of the contents provided in the paper's appendices.

- Appendix B contains the proof for Theorem 1.
- Appendix C describes the environments used in the experimental evaluation.

B Proofs

In this section, we provide the proof of Theorem 1.

We first relate the performance difference between the optimal policy in the target MDP and the policy that acts greedily with respect to $\Phi_T \widehat{U}^T \theta_S^\star$ to the estimation error in $\widehat{U}$, where $\widehat{U}$ is the output of Algorithm 2:

Lemma 1 (Performance Difference)**.** $\|V_T^\star - V_T^{\hat{\pi}}\|_\infty \leq 2H \|\widehat{U} - U_\star\|_{\max}$.

Here, $\|\cdot\|_{\max}$ denotes the max norm. To bound the estimation error $\|\widehat{U} - U_\star\|_{\max}$, we use the following per-row guarantee:

Lemma 2 (Estimation Error Bound)**.** *Let $M[i]$ denote the i-th row of a matrix M. Fix $i \in [d_S]$. Then, for all $n \geq \frac{6}{H\lambda_{\min}}(\log \frac{3d_T}{\delta})$, with probability at least $1-\delta$, the output $\widehat{U}$ of Algorithm 2 satisfies:*

$$\left\|\widehat{U}[i] - U_\star[i]\right\|_\Sigma \leq \sqrt{\frac{1}{nH}} \sqrt{d_T + 2\log\left(\frac{3}{\delta}\right) + 2\sqrt{d_T \log\left(\frac{3}{\delta}\right)}} + o\left(\sqrt{\frac{1}{nH}}\right).$$

Since Σ is positive semidefinite (PSD), we can relate the Σ-norm to the ℓ_∞ norm via:

$$\|x\|_\infty \leq \frac{\|x\|_\Sigma}{\sqrt{\lambda_{\min}}}. \tag{8}$$

Applying a union bound over all $i \in [d_S]$, with per-row failure probability set to δ/d_S, gives a high-probability bound on $\|U - U_\star\|_{\max}$. Substituting this into Lemma 1 concludes the proof.

Proof of Lemma 1. For any $s \in \mathcal{S}^T$,

$$\begin{aligned}
V_T^\star(s) - V_T^{\hat{\pi}}(s) &= Q_T^\star(s, \pi^\star(s)) - Q_T^\star(s, \hat{\pi}(s)) + Q_T^\star(s, \hat{\pi}(s)) - Q_T^{\hat{\pi}}(s, \hat{\pi}(s)) \\
&\leq Q_T^\star(s, \pi^\star(s)) - f(s, \pi^\star(s)) + f(s, \hat{\pi}(s)) - Q_T^\star(s, \hat{\pi}(s)) \\
&\quad + \mathbb{E}_{s' \sim \mathbb{P}^T(s, \hat{\pi}(s))}\left[V_T^\star(s') - V_T^{\hat{\pi}}(s')\right] \\
&\leq 2\|f - Q_T^\star\|_\infty + \mathbb{E}_{s' \sim \mathbb{P}^T(s, \hat{\pi}(s))}\left[V_T^\star(s') - V_T^{\hat{\pi}}(s')\right].
\end{aligned}$$

Unrolling this recursion for H steps,

$$\|V_T^\star - V_T^{\hat{\pi}}\|_\infty \leq 2H \|f - Q_T^\star\|_\infty.$$

Setting $f = \Phi_T U^\top \theta_S^\star$ and $Q_T^\star = \Phi_T U_\star^\top \theta_S^\star$ completes the proof.

Proof of Lemma 2. Note that the noise in Algorithm 2 is 1-subgaussian. Therefore, the result follows from Theorem 1 and Remark 9 in (Hsu et al., 2014), where Condition 1 is satisfied with $\rho_0 \leq \frac{1}{\sqrt{\lambda_{\min} d_T}}$, Condition 2 is satisfied with $\sigma = 1$, and Condition 3 is satisfied with $b_0 = 0$.

C Environments

ROTATEDSWIMMER. This task requires the agent to swim forward through a viscous fluid. We apply a 45° rotation to the (x, y) plane to modify the frame in which the agent's motion is observed. This affects both the heading of the swimmer and its linear velocity vector. The observation is updated accordingly by adjusting the front tip's angle and rotating the (x, y) velocity components.

ROTATEDHALFCHEETAH. This task requires the agent to run forward in a planar environment. The original environment operates in a 2D plane. We perform a 45° rotation on the (x, z) components of the agent's torso position and velocity. The joint states and angular velocities remain unchanged.

ROTATEDWALKER2D. This task requires the agent to walk forward while maintaining balance. Similar to HalfCheetah, we rotate the observation of the agent's torso height and linear velocity in the (x, z) plane by 45°. This alters how progress and vertical motion are perceived, while the actuator and sensor states are kept intact.

ROTATEDANT. In this task, the agent is a quadraped and must navigate a 2D surface using four legs. We rotate the global (x, y) velocity of the torso by 45°. When present, external contact forces are also rotated in the same plane. The result is a consistent shift in perceived motion direction across the ant's high-dimensional observation space.

Sensor Fusion. In the fusion environments, instead of observing a d_S-dimensional observation vector, the agent receives a $10 \times d_S$-dimensional observation. More concretely, for each source observation element i, the target task provides 10 measurements of the form: $x_i^{(j)} = w_j \cdot x_i + c_j$, for $j \in [10]$, where the weights w_j sum to 1 and the offsets c_j sum to 0. This simulates a sensor fusion scenario in which the undo map $u_\star$ sums the 10 measurements corresponding to each original dimension to recover the source observation.