Unsupervised Cross-Domain Transfer in Policy Gradient Reinforcement Learning via Manifold Alignment

Haitham Bou Ammar
Univ. of Pennsylvania
haithamb@seas.upenn.edu

Eric Eaton
Univ. of Pennsylvania
eeaton@cis.upenn.edu

Paul Ruvolo
Olin College of Engineering
paul.ruvolo@olin.edu

Matthew E. Taylor
Washington State Univ.
taylorm@eecs.wsu.edu

Abstract

The success of applying policy gradient reinforcement learning (RL) to difficult control tasks hinges crucially on the ability to determine a sensible initialization for the policy. Transfer learning methods tackle this problem by reusing knowledge gleaned from solving other related tasks. In the case of multiple task domains, these algorithms require an *inter-task mapping* to facilitate knowledge transfer across domains. However, there are currently no general methods to learn an inter-task mapping without requiring either background knowledge that is not typically present in RL settings, or an expensive analysis of an exponential number of inter-task mappings in the size of the state and action spaces.

This paper introduces an autonomous framework that uses unsupervised manifold alignment to learn inter-task mappings and effectively transfer samples between different task domains. Empirical results on diverse dynamical systems, including an application to quadrotor control, demonstrate its effectiveness for cross-domain transfer in the context of policy gradient RL.

Introduction

Policy gradient reinforcement learning (RL) algorithms have been applied with considerable success to solve high-dimensional control problems, such as those arising in robotic control and coordination (Peters & Schaal 2008). These algorithms use gradient ascent to tune the parameters of a policy to maximize its expected performance. Unfortunately, this gradient ascent procedure is prone to becoming trapped in local maxima, and thus it has been widely recognized that initializing the policy in a sensible manner is crucial for achieving optimal performance. For instance, one typical strategy is to initialize the policy using human demonstrations (Peters & Schaal 2006), which may be infeasible when the task cannot be easily solved by a human. This paper explores a different approach: instead of initializing the policy at random (i.e., *tabula rasa*) or via human demonstrations, we instead use transfer learning (TL) to initialize the policy for a new target domain based on knowledge from one or more source tasks.

In RL transfer, the source and target tasks may differ in their formulations (Taylor & Stone 2009). In particular, when the source and target tasks have different state and/or action spaces, an *inter-task mapping* (Taylor et al. 2007a) that describes the relationship between the two tasks is typically needed. This paper introduces a framework for autonomously learning an inter-task mapping for cross-domain transfer in policy gradient RL. First, we learn an inter-state mapping (i.e., a mapping between states in two tasks) using unsupervised manifold alignment. Manifold alignment provides a powerful and general framework that can discover a shared latent representation to capture intrinsic relations between different tasks, irrespective of their dimensionality. The alignment also yields an implicit inter-action mapping that is generated by mapping *tracking* states from the source to the target. Given the mapping between task domains, source task trajectories are then used to initialize a policy in the target task, significantly improving the speed of subsequent learning over an uninformed initialization.

This paper provides the following contributions. First, we introduce a novel unsupervised method for learning inter-state mappings using manifold alignment. Second, we show that the discovered subspace can be used to initialize the target policy. Third, our empirical validation conducted on four dissimilar and dynamically chaotic task domains (e.g., controlling a three-link cart-pole and a quadrotor aerial vehicle) shows that our approach can *a*) automatically learn an inter-state mapping across MDPs from the same domain, *b*) automatically learn an inter-state mapping across MDPs from *very different* domains, and *c*) transfer informative initial policies to achieve higher initial performance and reduce the time needed for convergence to near-optimal behavior.

Related Work

Learning an inter-task mapping has been of major interest in the transfer learning community because of its promise of autonomous transfer between very different tasks (Taylor & Stone 2009). However, the majority of existing work assumes that *a*) the source task and target task are similar enough that no mapping is needed (Banerjee & Stone 2007; Konidaris & Barto 2007), or *b*) an inter-task mapping is provided to the agent (Taylor et al. 2007a; Torrey et al. 2008). The main difference between these methods and this paper is that we are interested in *learning* a mapping between tasks.

There has been some recent work on learning such mappings. For example, mappings may be based on seman-

tic knowledge about state features between two tasks (Liu & Stone 2006), background knowledge about the range or type of state variables (Taylor et al. 2007b), or transition models for each possible mapping could be generated and tested (Taylor et al. 2008). However, there are currently no general methods to learn an inter-task mapping without requiring either background knowledge that is not typically present in RL settings, or an expensive analysis of an exponential number (in the size of the action and state variable sets) of inter-task mappings. We overcome these issues by automatically discovering high-level features and using them to transfer knowledge between agents without suffering from an exponential explosion.

In previous work, we used sparse coding, sparse projection, and sparse Gaussian processes to learn an inter-task mapping between MDPs with arbitrary variations (Bou Ammar et al. 2012). However, this previous work relied on a Euclidean distance correlation between source and target task triplets, which may fail for highly dissimilar tasks. Additionally, it placed restrictions on the inter-task mapping that reduced the flexibility of the learned mapping. In other related work, Bósci *et al.* (2013) use manifold alignment to assist in transfer. The primary differences with our work are that the authors *a*) focus on transferring models between different robots, rather than policies/samples, and *b*) rely on source and target robots that are qualitatively similar.

Background

Reinforcement Learning problems involve an agent choosing sequential actions to maximize its expected return. Such problems are typically formalized as a Markov decision process (MDP) $\mathcal{T} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_0, \mathcal{P}, r \rangle$, where $\mathcal{S}$ is the (potentially infinite) set of states, $\mathcal{A}$ is the set of actions that the agent may execute, $\mathcal{P}_0 : \mathcal{S} \rightarrow [0, 1]$ is a probability distribution over the initial state, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition probability function describing the task dynamics, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function measuring the performance of the agent. A policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is defined as a conditional probability distribution over actions given the current state. The agent's goal is to find a policy $\pi^\star$ which maximizes the average expected reward:

$$\pi^\star = \arg\max_\pi \mathrm{E}\left[\frac{1}{H}\sum_{t=1}^{H} r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \middle| \pi\right] \quad (1)$$
$$= \arg\max_\pi \int_{\mathbb{T}} p_\pi(\boldsymbol{\tau}) \mathcal{R}(\boldsymbol{\tau}) d\boldsymbol{\tau} \ ,$$

where $\mathbb{T}$ is the set of all possible trajectories with horizon H,

$$\mathcal{R}(\boldsymbol{\tau}) = \frac{1}{H}\sum_{t=1}^{H} r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \ , \text{and} \quad (2)$$

$$p_\pi(\boldsymbol{\tau}) = \mathcal{P}_0(\boldsymbol{s}_1) \prod_{t=1}^{H} \mathcal{P}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t) \, \pi(\boldsymbol{a}_t | \boldsymbol{s}_t) \ . \quad (3)$$

Policy Gradient methods (Sutton et al. 1999; Peters et al. 2005) represent the agent's policy π as a function defined over a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ of control parameters and a vector of state features given by the transformation $\boldsymbol{\Phi} : \mathcal{S} \rightarrow \mathbb{R}^m$. By substituting this parameterization of the control policy into Eqn. (2), we can compute the parameters of the optimal policy as $\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$, where $\mathcal{J}(\boldsymbol{\theta}) = \int_{\mathbb{T}} p_{\pi(\boldsymbol{\theta})}(\boldsymbol{\tau}) \mathcal{R}(\boldsymbol{\tau}) d\boldsymbol{\tau}$. To maximize $\mathcal{J}$, many policy gradient methods employ standard supervised function approximation to learn $\boldsymbol{\theta}$ by following an estimated gradient of a lower bound on the expected return of $\mathcal{J}(\boldsymbol{\theta})$.

Policy gradient algorithms have gained attention in the RL community in part due to their successful applications on real-world robotics (Peters et al. 2005). While such algorithms have a low computational cost per update, high-dimensional problems require many updates (by acquiring new rollouts) to achieve good performance. Transfer learning can reduce this data requirement and accelerate learning.

Since policy gradient methods are prone to becoming stuck in local maxima, it is crucial that the policy be initialized in a sensible fashion. A common technique (Peters & Schaal 2006; Argall et al. 2009) for policy initialization is to first collect demonstrations from a human controlling the system, then use supervised learning to fit policy parameters that maximize the likelihood of the human-demonstrated actions, and finally use the fitted parameters as the initial policy parameters for a policy gradient algorithm. While this approach works well in some settings, it is inapplicable in several common cases: *a*) when it is difficult to instrument the system in question so that a human can successfully perform a demonstration, *b*) when an agent is constantly faced with new tasks, making gathering human demonstrations for each new task impractical, or *c*) when the tasks in question cannot be intuitively solved by a human demonstrator.

The next section introduces a method for using transfer learning to initialize the parameters of a policy in a way that is not susceptible to these limitations. Our experimental results show that this method of policy initialization, when compared to random policy initialization, is able to not only achieve better initial performance, but also obtain a higher performing policy when run until convergence.

Policy Gradient Transfer Learning

Transfer learning aims to improve learning times and/or behavior of an agent on a new target task $\mathcal{T}^{(T)}$ by reusing knowledge from a solved source task $\mathcal{T}^{(S)}$. In RL settings, each task is described by an MDP: task $\mathcal{T}^{(S)} = \langle \mathcal{S}^{(S)}, \mathcal{A}^{(S)}, \mathcal{P}_0^{(S)}, \mathcal{P}^{(S)}, r^{(S)} \rangle$ and $\mathcal{T}^{(T)} = \langle \mathcal{S}^{(T)}, \mathcal{A}^{(T)}, \mathcal{P}_0^{(T)}, \mathcal{P}^{(T)}, r^{(T)} \rangle$. One way in which knowledge from a solved source task can be leveraged to solve the target task is by mapping optimal $\langle$state, action, next state$\rangle$ triples from the source task into the state and action spaces of the target task. Transferring optimal triples in this way allows us to both provide a better jumpstart and learning ability to the target agent, based on the source agent's ability.

While the preceding idea is attractive, complexities arise when the source and target tasks have different state and/or action spaces. In this case, one must define an inter-task mapping $\boldsymbol{\chi}$ in order to translate optimal triples from the source to the target task. Typically (Taylor & Stone 2009), $\boldsymbol{\chi}$ is defined by two sub-mappings: (1) an inter-state mapping $\boldsymbol{\chi}_{\mathcal{S}}$ and (2) an inter-action mapping $\boldsymbol{\chi}_{\mathcal{A}}$.

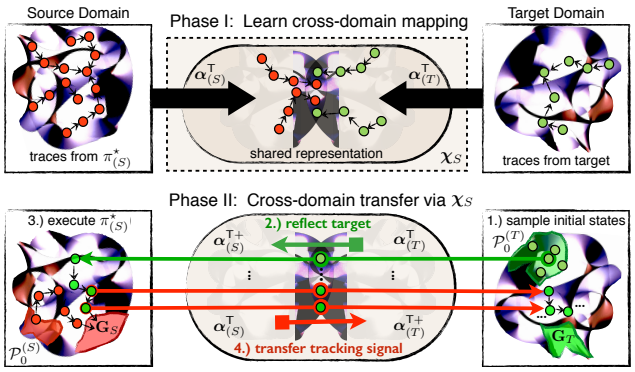


Figure 1: Transfer is split into two phases: (I) learning the inter-state mapping $\chi_{\mathcal{S}}$ via manifold alignment, and (II) initializing the target policy via mapping the source task policy.

By adopting an RL framework where policies are state-feedback controllers, we show that we can use optimal state trajectories from the source task to intelligently initialize a control policy in the target task, without needing to explicitly construct an inter-action mapping. We accomplish this by learning a (pseudo-invertible) inter-state mapping between the state spaces of a pair of tasks using manifold alignment, which can then be used to transfer optimal sequences of states to the target. The fact that our algorithm does not require learning an explicit inter-action mapping significantly reduces its computational complexity.

Our approach consists of two phases (Figure 1). First, using traces gathered in the source and target tasks, we learn an inter-state mapping $\chi_{\mathcal{S}}$ using manifold alignment ("Phase I" in Figure 1). To perform this step, we adapt the Unsupervised Manifold Alignment (UMA) algorithm (Wang & Mahadevan 2009), as detailed in the next section. Second, we use $\chi_{\mathcal{S}}$ to project state trajectories from the source to the target task ("Phase II" in Figure 1). These projected state trajectories define a set of a tracking trajectories for the target task that allow us to perform one step of policy gradient improvement in the target task. This policy improvement step intelligently initializes the target policy, which results in superior learning performance than starting from a randomly initialized policy, as shown in our experiments. Although we focus on policy gradient methods, our approach could easily be adapted to other policy search methods (e.g., PoWER, REPS, etc.; see Kober et al. 2013).

Learning an Inter-State Mapping

Unsupervised Manifold Alignment (UMA) is a technique that efficiently discovers an alignment between two datasets (Wang & Mahadevan 2009). UMA was developed to align datasets for knowledge transfer between two supervised learning tasks. Here, we adapt UMA to an RL setting by aligning source and target task state spaces with potentially different dimensions m_S and m_T. To learn $\chi_{\mathcal{S}}$ relating $\mathcal{S}^{(S)}$ and $\mathcal{S}^{(T)}$, trajectories of states in the source task, $\boldsymbol{\tau}^{\star}_{(S)} = \left\{\mathbf{s}_1^{(i),(S)\star}, \ldots, \mathbf{s}_{H_S}^{(i),(S)\star}\right\}_{i=1}^{n_S}$, are obtained by following $\pi^{\star}_{(S)}$, and trajectories of states in the target task, $\boldsymbol{\tau}_{(T)} = \left\{s_1^{(j),(T)}, \ldots, s_{H_T}^{(j),(T)}\right\}_{j=1}^{n_T}$, are obtained by utilizing $\pi_{(T)}$, which is initialized using randomly selected policy parameters. For simplicity of exposition, we assume that trajectories in the source domain have length H_S and those in the target domain have length H_T; however, our algorithm is capable of handling variable-length trajectories. We are interested in the setting where data is scarcer in the target task than in the source task (i.e., $n_T \ll n_S$).

Given trajectories from both the source and target tasks, we flatten the trajectories (i.e., we treat the states as unordered) and then apply the task-specific state transformation to obtain two sets of state feature vectors, one for the source task and one for the target task. Specifically, we create the following sets of points:

$$\mathbf{X}^{(S)} = \Big\{\Phi^{(S)}\left(\mathbf{s}_1^{(1)(S)\star}\right), \ldots, \Phi^{(S)}\left(\mathbf{s}_{H_S}^{(1)(S)\star}\right), \Phi^{(S)}\left(\mathbf{s}_1^{(n_S)(S)\star}\right), \ldots, \Phi^{(S)}\left(\mathbf{s}_{H_S}^{(n_S)(S)\star}\right)\Big\}$$

$$\mathbf{X}^{(T)} = \Big\{\Phi^{(T)}\left(\mathbf{s}_1^{(1)(T)}\right), \ldots, \Phi^{(T)}\left(\mathbf{s}_{H_T}^{(1)(T)}\right), \Phi^{(T)}\left(\mathbf{s}_1^{(n_T)(T)}\right), \ldots, \Phi^{(T)}\left(\mathbf{s}_{H_T}^{(n_T)(T)}\right)\Big\} .$$

Given $\mathbf{X}^{(S)} \in \mathbb{R}^{m_S \times (H_S \times n_S)}$, $\mathbf{X}^{(T)} \in \mathbb{R}^{m_T \times (H_T \times n_T)}$, we can apply the UMA algorithm (Wang & Mahadevan 2009) with minimal modification, as described next.

Unsupervised Manifold Alignment (UMA) The first step of applying UMA to learn the inter-state mapping is to represent each transformed state in both the source and target tasks in terms of its local geometry. We use the notation $\mathbf{R}_{\mathbf{x}_i^{(S)}} \in \mathbb{R}^{(k+1)\times(k+1)}$ to refer to the matrix of pairwise Euclidean distances among the k-nearest neighbors of $\mathbf{x}_i^{(S)} \in \mathbf{X}^{(S)}$. Similarly, $\mathbf{R}_{\mathbf{x}_j^{(T)}}$ refers to the equivalent matrix of distances for the k-nearest neighbors of $\mathbf{x}_j^{(T)} \in \mathbf{X}^{(T)}$.

The relations between local geometries in $\mathbf{X}^{(S)}$ and $\mathbf{X}^{(T)}$ are represented by the matrix $\mathbf{W} \in \mathbb{R}^{(n_S \times H_S)\times(n_T \times H_T)}$ with $w_{i,j} = \exp\left\{-\text{dist}\left(\mathbf{R}_{\mathbf{x}_i^{(S)}}, \mathbf{R}_{\mathbf{x}_j^{(T)}}\right)\right\}$ and distance metric

$$\text{dist}\left(\mathbf{R}_{\mathbf{x}_i^{(S)}}, \mathbf{R}_{\mathbf{x}_j^{(T)}}\right) = \min_{1\le h\le k!}\left[\min\left(\left\|\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h - \gamma_1\mathbf{R}_{\mathbf{x}_i^{(S)}}\right\|_{\text{F}}, \left\|\mathbf{R}_{\mathbf{x}_i^{(S)}} - \gamma_2\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h\right\|_{\text{F}}\right)\right] . \quad (4)$$

We use the notation $\wr\cdot\wr_h$ to denote the h^{th} variant of the $k!$ permutations of the rows and columns of the input matrix, $||\cdot||_{\text{F}}$ is the Frobenius norm, and γ_1 and γ_2 are defined as:

$$\gamma_1 = \frac{\text{tr}\left(\mathbf{R}_{\mathbf{x}_i^{(S)}}^{\mathsf{T}}\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h\right)}{\text{tr}\left(\mathbf{R}_{\mathbf{x}_i^{(S)}}^{\mathsf{T}}\mathbf{R}_{\mathbf{x}_i^{(S)}}\right)} \qquad \gamma_2 = \frac{\text{tr}\left(\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h^{\mathsf{T}}\mathbf{R}_{\mathbf{x}_i^{(S)}}\right)}{\text{tr}\left(\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h^{\mathsf{T}}\wr\mathbf{R}_{\mathbf{x}_j^{(T)}}\wr_h\right)} .$$

To align the manifolds, UMA computes the joint Laplacian

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{\mathbf{X}^{(S)}} + \mu\mathbf{\Gamma}^{(1)} & -\mu\mathbf{\Gamma}^{(2)} \\ -\mu\mathbf{\Gamma}^{(3)} & \mathbf{L}_{\mathbf{X}^{(T)}} + \mu\mathbf{\Gamma}^{(4)} \end{pmatrix} \quad (5)$$

with diagonal matrices $\mathbf{\Gamma}^{(1)} \in \mathbb{R}^{(n_S \times H_S)\times(n_S \times H_S)}$ and $\mathbf{\Gamma}^{(4)} \in \mathbb{R}^{(n_T \times H_T)\times(n_T \times H_T)}$, where $\mathbf{\Gamma}^{(1)}_{i,i} = \sum_j w_{i,j}$ and $\mathbf{\Gamma}^{(4)}_{j,j} = \sum_i w_{i,j}$. The matrices $\mathbf{\Gamma}^{(2)} \in \mathbb{R}^{(n_S \times H_S)\times(n_T \times H_T)}$ and $\mathbf{\Gamma}^{(3)} \in \mathbb{R}^{(n_T \times H_T)\times(n_S \times H_S)}$ join the two manifolds with $\mathbf{\Gamma}^{(2)}_{i,j} = w_{i,j}$ and $\mathbf{\Gamma}^{(3)}_{i,j} = w_{j,i}$.

Additionally, the non-normalized Laplacians $\mathbf{L}_{\mathbf{X}^{(S)}}$ and $\mathbf{L}_{\mathbf{X}^{(T)}}$ are defined as: $\mathbf{L}_{\mathbf{X}^{(S)}} = \mathbf{D}_{\mathbf{X}^{(S)}} - \mathbf{W}_{\mathbf{X}^{(S)}}$ and $\mathbf{L}_{\mathbf{X}^{(T)}} = \mathbf{D}_{\mathbf{X}^{(T)}} - \mathbf{W}_{\mathbf{X}^{(T)}}$, where $\mathbf{D}_{\mathbf{X}^{(S)}} \in \mathbb{R}^{(n_S \times H_S)\times(n_S \times H_S)}$ is a diagonal matrix with $\mathbf{D}^{(i,i)}_{\mathbf{X}^{(S)}} = \sum_j w^{(S)}_{i,j}$ and, similarly, $\mathbf{D}^{(i,i)}_{\mathbf{X}^{(T)}} = \sum_j w^{(T)}_{i,j}$. The matrices $\mathbf{W}^{(S)}$ and $\mathbf{W}^{(T)}$ represent the similarity in the source and target task state spaces respectively and can be computed similar to $\mathbf{W}$.

To join the manifolds, UMA first defines two matrices:

$$\mathbf{Z} = \begin{pmatrix} \boldsymbol{\tau}^{\star}_{(S)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\tau}_{(T)} \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} \mathbf{D}_{\mathcal{S}^{(S)}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\mathcal{S}^{(T)}} \end{pmatrix} . \quad (6)$$

Given $\mathbf{Z}$ and $\mathbf{D}$, UMA computes optimal projections to reduce the dimensionality of the joint structure by taking the d minimum eigenvectors $\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_d$ of the generalized eigenvalue decomposition $\mathbf{ZLZ}^T\boldsymbol{\zeta} = \lambda\mathbf{ZDZ}^T\boldsymbol{\zeta}$. The optimal projections $\boldsymbol{\alpha}_{(S)}$ and $\boldsymbol{\alpha}_{(T)}$ are then given as the first d_1 and d_2 rows of $[\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_d]$, respectively.

Given the embedding discovered by UMA, we can then define the inter-state mapping as:

$$\boldsymbol{\chi}_{\mathcal{S}}[\cdot] = \boldsymbol{\alpha}^{\mathsf{T}+}_{(T)}\boldsymbol{\alpha}^{\mathsf{T}}_{(S)}[\cdot] \ . \quad (7)$$

The inverse of the inter-state mapping (to project target states to the source task) can be determined by taking the pseudo-inverse of Eqn. (7), yielding $\boldsymbol{\chi}^{+}_{\mathcal{S}}[\cdot] = \boldsymbol{\alpha}^{\mathsf{T}+}_{(S)}\boldsymbol{\alpha}^{\mathsf{T}}_{(T)}[\cdot]$.

Intuitively, this approach aligns the important regions of the source task's state space (sampled based on optimal source trajectories) with the state space explored so far in the target task. Although actions were ignored in constructing the manifolds, the aligned representation implicitly captures local state transition dynamics within each task (since the states came from trajectories), providing a mechanism to transfer trajectories between tasks, as we describe next.

Policy Transfer and Improvement

Next, we discuss the procedure for initializing the target policy, $\pi_{(T)}$. We consider a model-free setting in which the policy is linear over a set of (potentially) non-linear state feature functions modulated by Gaussian noise (where the magnitude of the noise balances exploration and exploitation). Specifically, we can write the source and target policies as:

$$\pi^{\star}_{(S)}\left(\boldsymbol{s}^{(S)}_t\right) = \mathbf{\Phi}^{(S)}\left(\boldsymbol{s}^{(S)}_t\right)^{\mathsf{T}}\boldsymbol{\theta}^{(S)\star} + \boldsymbol{\epsilon}^{(S)}_t$$
$$\pi_{(T)}\left(\boldsymbol{s}^{(T)}_t\right) = \mathbf{\Phi}^{(T)}\left(\boldsymbol{s}^{(T)}_t\right)^{\mathsf{T}}\boldsymbol{\theta}^{(T)} + \boldsymbol{\epsilon}^{(T)}_t \ ,$$

where $\boldsymbol{\epsilon}^{(S)}_t \sim \mathcal{N}\left(0, \Sigma^{(S)}\right)$ and $\boldsymbol{\epsilon}^{(T)}_t \sim \mathcal{N}\left(0, \Sigma^{(T)}\right)$.

To initialize $\pi_{(T)}$, we first sample m initial target trajectories $\mathcal{D}^{(T)} = \left\{\boldsymbol{\tau}^{(T)}_i\right\}_{i=1}^{m}$ from the target task using a randomly initialized policy (these can be newly sampled trajectories or simply the ones used to do the initial manifold alignment step). Next, we map the set of initial states in $\mathcal{D}^{(T)}$ to the source task using $\boldsymbol{\chi}^{+}_{\mathcal{S}}$. We then run the optimal source policy starting from each of these mapped initial states to produce a set of m optimal state trajectories in the source task. Finally, the resulting state trajectories are mapped back to the target task using $\boldsymbol{\chi}_{\mathcal{S}}$ to generate a set of reflected state-trajectories in the target task, $\tilde{\mathcal{D}}^{(T)} = \left\{\tilde{\boldsymbol{\tau}}^{(T)}_i\right\}_{i=1}^{m}$. For clarity, we assume that all trajectories are of length H; however, this is not a fundamental limitation of our algorithm.

We define the following transfer cost function:

$$\mathcal{J}_{\mathcal{T}^{(S)}\rightarrow\mathcal{T}^{(T)}}\left(\boldsymbol{\theta}^{(T)}\right) = \sum_{i=1}^{m} p_{\boldsymbol{\theta}^{(T)}}\left(\boldsymbol{\tau}^{(T)}_i\right)\hat{\mathcal{R}}_{(T)}\left(\boldsymbol{\tau}^{(T)}_i, \tilde{\boldsymbol{\tau}}^{(T)}_i\right) \quad (8)$$

where $\hat{\mathcal{R}}_{(T)}$ is a cost function that penalizes deviations between the initial sampled trajectories in the target task and the reflected optimal trajectories:

$$\hat{\mathcal{R}}_{(T)}\left(\boldsymbol{\tau}^{(T)}, \tilde{\boldsymbol{\tau}}^{(T)}\right) = \frac{1}{H}\sum_{t=1}^{H}\left|\left|\boldsymbol{s}^{(T)}_t - \tilde{\boldsymbol{s}}^{(T)}_t\right|\right|^2_2 \ . \quad (9)$$

Minimizing, Eqn. (8) is equivalent to attaining a target policy parameterization $\boldsymbol{\theta}^{(T)}$ such that $\pi_{(T)}$ follows the reflected trajectories $\tilde{\mathcal{D}}^{(T)}$. Further, Eqn. (8) is in exactly the form required to apply standard off-the-shelf policy gradient algorithms to minimize the transfer cost. The Manifold Alignment Cross-Domain Transfer for Policy Gradients (MAXDT-PG) framework is detailed[1] in Algorithm 1.

Special Cases

Our work can be seen as an extension of the simpler model-based case with a linear-quadratic regulator (LQR) (Bemporad et al. 2002) policy, which is derived and explained in the online appendix[2] accompanying this paper. Although the assumptions made by the model-based case seem restrictive, the analysis in the appendix covers a wide range of applications. These, for example, include: *a*) the case in which a dynamical model is provided beforehand, or *b*) the case in which model-based RL algorithms are adopted (see Buşoniu et al. 2010). In the main paper, however, we consider the more general model-free case.

Experiments and Results

To assess MAXDT-PG's performance, we conduced experiments on transfer both between tasks in the same domain as well as between tasks in different domains. Also, we studied

[1] Lines 9-11 of Algorithm 1 require interaction with the target domain (or a simulator) for acquiring the optimal policy. Such an assumption is common to policy gradient methods, where at each iteration, data is gathered and used to iteratively improve the policy.

[2] The online appendix is available on the authors' websites.

Algorithm 1 Manifold Alignment Cross-Domain Transfer for Policy Gradients (MAXDT-PG)

Inputs: Source and target tasks $\mathcal{T}^{(S)}$ and $\mathcal{T}^{(T)}$, optimal source policy $\pi^{\star}_{(S)}$, # source and target traces n_S and n_T, # nearest neighbors k, # target rollouts z_T, initial # of target states m.

Learn $\boldsymbol{\chi_{\mathcal{S}}}$:

1: Sample n_S optimal source traces, $\boldsymbol{\tau}^{\star}_{(S)}$, and n_T random target traces, $\boldsymbol{\tau}_{(T)}$

2: Using the modified UMA approach, learn $\boldsymbol{\alpha}_{(S)}$ and $\boldsymbol{\alpha}_{(T)}$ to produce $\boldsymbol{\chi_{\mathcal{S}}} = \boldsymbol{\alpha}^{\mathsf{T}+}_{(T)}\boldsymbol{\alpha}^{\mathsf{T}}_{(S)}[\cdot]$

Transfer & Initialize Policy:

3: Collect m initial target states $\boldsymbol{s}^{(T)}_1 \sim \mathcal{P}^{(T)}_0$

4: Project these m states to the source by applying $\boldsymbol{\chi}^{+}_{\mathcal{S}}[\cdot]$

5: Apply the optimal source policy $\pi^{\star}_{(S)}$ on these projected states to collect $\mathcal{D}^{(S)} = \left\{\boldsymbol{\tau}^{(S)}_{(i)}\right\}^{m}_{i=1}$

6: Project the samples in $\mathcal{D}^{(S)}$ to the target using $\boldsymbol{\chi_{\mathcal{S}}}[\cdot]$ to produce tracking target traces $\tilde{\mathcal{D}}^{(T)}$

7: Compute tracking rewards using Eqn. (9)

8: Use policy gradients to minimize Eqn. (8), yielding $\boldsymbol{\theta}^{(0)}_{(T)}$

Improve Policy:

9: Start with $\boldsymbol{\theta}^{(0)}_{(T)}$ and sample z_T target rollouts

10: Follow policy gradients (e.g., episodic REINFORCE) but using target rewards $\mathcal{R}^{(T)}$

11: Return optimal target policy parameters $\boldsymbol{\theta}^{\star}_{(T)}$

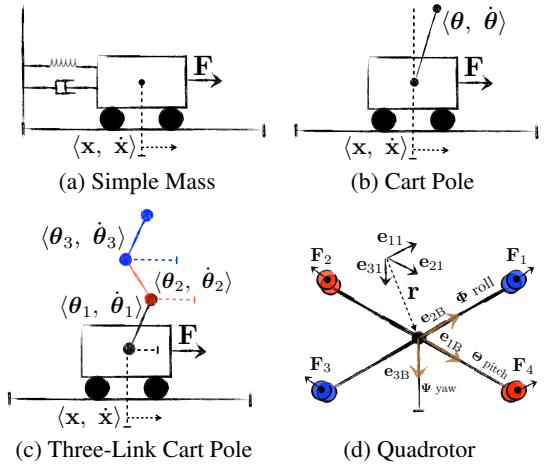
the robustness of the learned mapping by varying the number of source and target samples used for transfer and measuring the resultant target task performance. In all cases we compared the performance of MAXDT-PG to standard policy gradient learners. Our results show that MAXDT-PG was able to: *a*) learn a valid inter-state mapping with relatively little data from the target task, and *b*) effectively transfer between tasks from either the same or different domains.

Dynamical System Domains

We tested MAXDT-PG and standard policy gradient learning on four dynamical systems (Figure 2). On all systems, the reward function was based on two factors: *a*) penalizing states far from the goal state, and *b*) penalizing high forces (actions) to encourage smooth, low-energy movements.

Simple Mass Spring Damper (SM): The goal with the SM is to control the mass at a specified position with zero velocity. The system dynamics are described by two state-variables that represent the mass position and velocity, and a single force $\boldsymbol{F}$ that acts on the cart in the $\mathbf{x}$ direction.

Cart Pole (CP): The goal is to swing up and then balance the pole vertically. The system dynamics are described via a four-dimensional state vector $\langle \mathbf{x},\ \dot{\mathbf{x}},\ \boldsymbol{\theta},\ \dot{\boldsymbol{\theta}} \rangle$, representing the position, velocity of the cart, and the angle and angular velocity of the pole, respectively. The actions consist of a force that acts on the cart in the $\mathbf{x}$ direction.



(a) Simple Mass (b) Cart Pole

(c) Three-Link Cart Pole (d) Quadrotor

Figure 2: Dynamical systems used in the experiments.

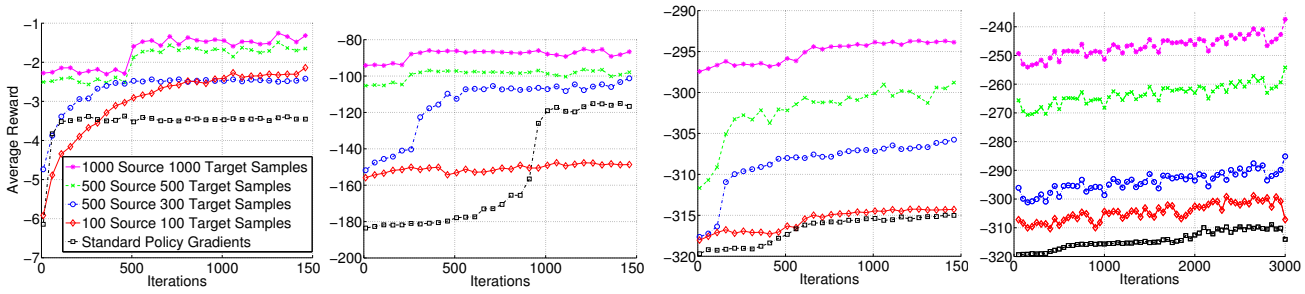
Three-Link Cart Pole (3CP): The 3CP dynamics are described via an eight-dimensional state vector $\langle \mathbf{x},\ \dot{\mathbf{x}},\ \boldsymbol{\theta}_1,\ \dot{\boldsymbol{\theta}}_1,\ \boldsymbol{\theta}_2,\ \dot{\boldsymbol{\theta}}_2,\ \boldsymbol{\theta}_3,\ \dot{\boldsymbol{\theta}}_3 \rangle$, where $\mathbf{x}$ and $\dot{\mathbf{x}}$ describe the position and velocity of the cart and $\boldsymbol{\theta}_j$ and $\dot{\boldsymbol{\theta}}_j$ represent the angle and angular velocity of the j^{th} link. The system is controlled by applying a force $\boldsymbol{F}$ to the cart in the $\mathbf{x}$ direction, with the goal of balancing the three poles upright.

Quadrotor (QR): The system dynamics were adopted from a simulator validated on real quadrotors (Bouabdallah 2007; Voos & Bou Ammar 2010), and are described via three angles and three angular velocities in the body frame (i.e., $\boldsymbol{e}_{1\mathrm{B}}$, $\boldsymbol{e}_{2\mathrm{B}}$, and $\boldsymbol{e}_{3\mathrm{B}}$). The actions consist of four rotor torques $\{\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3, \boldsymbol{F}_4\}$. Each task corresponds to a different quadrotor configuration (e.g., different armature lengths, etc.), and the goal is to stabilize the different quadrotors.

Same-Domain Transfer

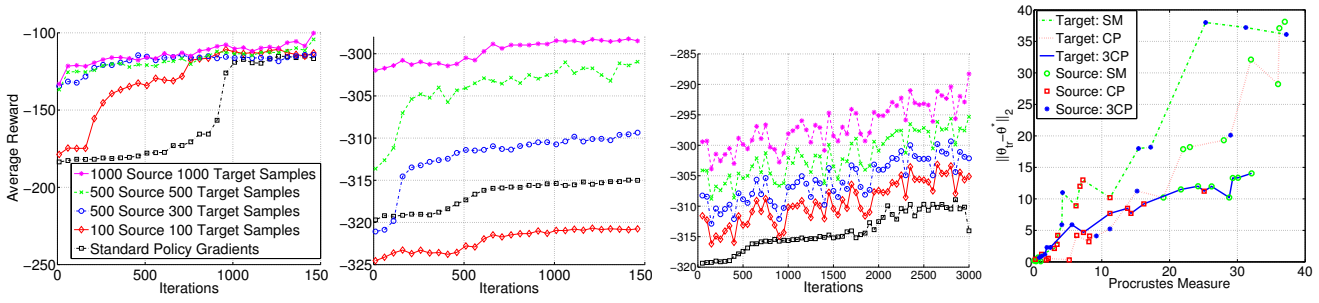
We first evaluate MAXDT-PG on same-domain transfer. Within each domain, we can obtain different tasks by varying the system parameters (e.g., for the SM system we varied mass M, spring constant K, and damping constant b) as well as the reward functions. We assessed the performance of using the transferred policy from MAXDT-PG versus standard policy gradients by measuring the average reward on the target task vs. the amount of learning iterations in the target. We also examined the robustness of MAXDT-PG's performance based on the number of source and target samples used to learn $\boldsymbol{\chi_{\mathcal{S}}}$. Rewards were averaged over 500 traces collected from 150 initial states. Due to space constraints, we report same-domain transfer results here; details of the tasks and experimental procedure can be found in the appendix[2].

Figure 3 shows MAXDT-PG's performance using varying numbers of source and target samples to learn $\boldsymbol{\chi_{\mathcal{S}}}$. These results reveal that transfer-initialized policies outperform standard policy gradient initialization. Further, as the number of samples used to learn $\boldsymbol{\chi_{\mathcal{S}}}$ increases, so does both the initial and final performance in all domains. All initializations result in equal per-iteration computational cost. Therefore, MAXDT-PG both improves sample complexity and reduces wall-clock learning time.



(a) Simple Mass (b) Cart Pole (c) Three-Link Cart Pole (d) Quadrotor

Figure 3: Same-domain transfer results. All plots share the same legend and vertical axis label.



(a) Simple Mass to Cart Pole (b) Cart Pole to Three-Link CP (c) Cart Pole to Quadrotor (d) Alignment Quality vs Transfer

Figure 4: Cross-domain transfer results. Plots (a)–(c) depict target task performance, and share the same legend and axis labels. Plot (d) shows the correlation between manifold alignment quality (Procrustes metric) and quality of the transferred knowledge.

Cross-Domain Transfer

Next, we consider the more difficult problem of cross-domain transfer. The experimental setup is identical to the same-domain case with the crucial difference that the state and/or action spaces were different for the source and the target task (since the tasks were from different domains). We tested three cross-domain transfer scenarios: simple mass to cart pole, cart pole to three-link cart pole, and cart pole to quadrotor. In each case, the source and target task have different numbers of state variables and system dynamics. Details of these experiments are available in the appendix[2].

Figure 4 shows the results of cross-domain transfer, demonstrating that MAXDT-PG can achieve successful transfer between different task domains. These results reinforce the conclusions of the same-domain transfer experiments, showing that *a*) transfer-initialized policies outperform standard policy gradients, even between different task domains and *b*) initial and final performance improves as more samples are used to learn $\chi_{\mathcal{S}}$.

We also examined the correlation between the quality of the manifold alignment, as assessed by the Procrustes metric (Goldberg & Ritov 2009), and the quality of the transferred knowledge, as measured by the distance between the transferred ($\boldsymbol{\theta}_{\text{tr}}$) and the optimal ($\boldsymbol{\theta}^{\star}$) parameters (Figure 4(d)). On both measures, smaller values indicate better quality. Each data point represents a transfer scenario between two different tasks, from either SM, CP, or 3CP; we did not consider quadrotor tasks due to the required simulator time. Although we show that the Procrustes measure is positively correlated with transfer quality, we hesitate to recommend it as a predictive measure of transfer performance. In our approach, the cross-domain mapping is not guaranteed to be orthogonal, and therefore the Procrustes measure is not theoretically guaranteed to accurately measure the quality of the global embedding (i.e., Goldberg and Ritov's (2009) Corollary 1 is not guaranteed to hold), but the Procrustes measure still appears correlated with transfer quality in practice.

We can conclude that MAXDT-PG is capable of: *a*) automatically learning an inter-state mapping, and *b*) effectively transferring between different domain systems. Even when the source and target tasks are highly dissimilar (e.g., cart pole to quadrotor), MAXDT-PG is capable of successfully providing target policy initializations that outperform state-of-the-art policy gradient techniques.

Conclusion

We introduced MAXDT-PG, a technique for autonomous transfer between policy gradient RL algorithms. MAXDT-PG employs unsupervised manifold alignment to learn an inter-state mapping, which is then used to transfer samples and initialize the target task policy. MAXDT-PG's performance was evaluated on four dynamical systems, demonstrating that MAXDT-PG is capable of improving both an agent's initial and final performance relative to using policy gradient algorithms without transfer, even across different domains.

Acknowledgements

This research was supported by ONR grant #N00014-11-1-0139, AFOSR grant #FA8750-14-1-0069, and NSF grant IIS-1149917. We thank the anonymous reviewers for their helpful feedback.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Banerjee, B., and Stone, P. 2007. General game learning using knowledge transfer. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 672–677.

Bemporad, A.; Morari, M.; Dua, V.; and Pistikopoulos, E. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica* 38(1):3–20.

Bócsi, B.; Csato, L.; and Peters, J. 2013. Alignment-based transfer learning for robot models. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

Bou Ammar, H.; Taylor, M.; Tuyls, K.; Driessens, K.; and Weiss, G. 2012. Reinforcement learning transfer via sparse coding. In *Proceedings of the 11th Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Bouabdallah, S. 2007. *Design and Control of Quadrotors with Application to Autonomous Flying*. Ph.D. Dissertation, École polytechnique fédérale de Lausanne.

Buşoniu, L.; Babuška, R.; De Schutter, B.; and Ernst, D. 2010. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton, Florida: CRC Press.

Goldberg, Y.; and Ritov, Y. 2009. Local Procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Machine Learning* 77(1): 1–25.

Kober, J.; Bagnell, A.; and Peters, J. 2013. Reinforcement learning in robotics: a survey. *International Journal of Robotics Research* 32(11): 1238–1274.

Konidaris, G., and Barto, A. 2007. Building portable options: Skill transfer in reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 895–900.

Liu, Y., and Stone, P. 2006. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 415–20.

Peters, J., and Schaal, S. 2006. Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2219–2225.

Peters, J., and Schaal, S. 2008. Natural actor-critic. *Neurocomputing* 71(7-9):1180–1190.

Peters, J.; Vijayakumar, S.; and Schaal, S. 2005. Natural actor-critic. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, 280–291. Springer.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems* 1057–1063.

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

Taylor, M. E.; Kuhlmann, G.; and Stone, P. 2008. Autonomous transfer for reinforcement learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 283–290.

Taylor, M. E.; Stone, P.; and Liu, Y. 2007. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research* 8(1):2125–2167.

Taylor, M. E.; Whiteson, S.; and Stone, P. 2007. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*.

Torrey, L.; Shavlik, J.; Walker, T.; and Maclin, R. 2008. Relational macros for transfer in reinforcement learning. In Blockeel, H.; Ramon, J.; Shavlik, J.; and Tadepalli, P., eds., *Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 254–268.

Voos, H., and Bou Ammar, H. 2010. Nonlinear tracking and landing controller for quadrotor aerial robots. In *Proceedings of the IEEE International Conference on Control Applications (CCA)*, 2136–2141.

Wang, C., and Mahadevan, S. 2009. Manifold alignment without correspondence. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1273–1278. Morgan Kaufmann.