BeigeMaps: Behavioral Eigenmaps for Reinforcement Learning from Images

Sandesh Adhikary [1] Anqi Li [1 2] Byron Boots [1]

Abstract

Training reinforcement learning (RL) agents directly from high-dimensional image observations continues to be a challenging problem. Recent line of work on behavioral distances proposes to learn representations that encode behavioral similarities quantified by the bisimulation metric. By learning an isometric mapping to a lower dimensional space that preserves this metric, such methods attempt to learn representations that group together functionally similar states. However, such an isometric mapping may not exist, making the learning objective ill-defined. We propose an alternative objective that allows distortions in long-range distances, while preserving *local* metric structure – inducing representations that highlight natural clusters in the state space. This leads to new representations, which we term Behavioral Eigenmaps (BeigeMaps), corresponding to the eigenfunctions of similarity kernels induced by behavioral distances. We empirically demonstrate that when added as a drop-in modification, BeigeMaps improve the policy performance of prior behavioral distance based RL algorithms.

1. Introduction

Reinforcement learning (RL) from image observations holds great promise as it enables learning control policies for domains where obtaining low-dimensional state information is expensive or impossible (Yarats et al., 2019; Yen-Chen et al., 2020). However, image observations are typically high-dimensional and contain redundant or task-irrelevant information. Directly applying RL on image observations often results in poor sample efficiency and weak generalization capability (Laskin et al., 2020a; Zhang et al., 2020).

Central to this challenge is the problem of representation learning (Jaderberg et al., 2016; Laskin et al., 2020b; Zhang et al., 2020; Stooke et al., 2021). The goal of representation learning is to map high-dimensional observations, e.g. images, to low-dimensional representations, facilitating the learning of downstream policies and/or value functions. The representation can be learned a priori (Lange & Riedmiller, 2010; Lange et al., 2012) or jointly with the policy (Yarats et al., 2019; Zhang et al., 2020; Lee et al., 2020).

Recently, a set of representation learning approaches has been proposed using behavioral distances (Zhang et al., 2020; Castro et al., 2021; Kemertas & Aumentado-Armstrong, 2021; Castro et al., 2023; Chen & Pan, 2022). These distances are variants and relaxations of the bisimulation metric (Ferns et al., 2011; Castro, 2020) and capture state-differences based on expected returns under certain policies. These approaches not only encourage the low-dimensional representation to extract task-relevant information (Zhang et al., 2020), but also induce value-based metric structure in the representation space (Castro, 2020; Zhang et al., 2020). It has been shown that value function learning can benefit from such metric shaping, as states with similar values are grouped together in the representation space (Castro et al., 2021; 2023; Zhang et al., 2020).

Although prior approaches have proposed different behavioral distances, with considerations for computational complexity (Castro et al., 2021), training stability (Kemertas & Aumentado-Armstrong, 2021), approximation error (Chen & Pan, 2022), etc., they follow the same recipe when encoding the behavioral metric structure into the representation space. Namely, learning a representation mapping such that the low-dimensional representation space is *isometric*, i.e., *globally* distance preserving, to the space defined by a given behavioral distance.

Unfortunately, such an objective can be ill-defined – there may not exist a low-dimensional space with a metric that exactly matches the behavioral distance (Bourgain, 1985; Linial et al., 1995; Belkin & Niyogi, 2001; Castro et al., 2023). This issue may undermine the quality of the learned representation and induce instability in training since behavioral distances are computed through bootstrapping (Van Hasselt et al., 2018).

This raises a natural question: if isometry is ill-defined, is there a certain type of *acceptable* distortion that still pre-

[1]Computer Science and Engineering, University of Washington, Seattle, WA (USA) [2]NVIDIA; Work done while AL was affiliated with the University of Washington. Correspondence to: Sandesh Adhikary <adhikary@cs.washington.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

serves important geometric structure defined by the behavioral distance? We hypothesize that a good trade-off is to allow distortion in long-range distances while preserving the *local* geometric structure given by states that are closer together in terms of behavioral distance. This proposal is further motivated by the fact that locality preserving maps are known to emphasize natural clusters in data (Belkin & Niyogi, 2001). Since value-based state aggregation is a key motivation for behavioral distances (Ferns et al., 2011; Kemertas & Jepson, 2022), it may be beneficial to use locality preserving embeddings that are inherently better suited for this purpose.

We propose representations corresponding to the top[1] eigenfunctions of kernels defined with respect to behavioral distances, which we refer to as **B**ehavioral **Eigen****maps** (BeigeMaps). These eigenmaps correspond to the simultaneous optimal solution for locality preservation (Belkin & Niyogi, 2001) and state space partitioning (Von Luxburg, 2007). We demonstrate that the BeigeMap representation can be used as a drop-in modification of existing behavioral distance algorithms, and that it improves the policy performance of these algorithms when evaluated on the DeepMind Control Suite (Tassa et al., 2018).

2. Related Work

Representation Learning for RL Despite early success of RL with image inputs (Mnih et al., 2015), learning visual policies relying only on the return maximizing RL objective is challenging in domains with less structured visual input (Zhang et al., 2020; Lamb et al., 2022). Representation learning for RL seeks to improve sample efficiency (Castro et al., 2021; Laskin et al., 2020a; Yarats et al., 2021a), robustness (Zhang et al., 2020; Lamb et al., 2022), and generalization capability (Zhang et al., 2020; Laskin et al., 2020a) by learning a good, often low dimensional, representation.

A group of representation learning for RL approaches follows largely from self-supervised learning on visual tasks. Representations are shaped through auxiliary tasks such as reconstructing image observations (Lange & Riedmiller, 2010; Lange et al., 2012; Hafner et al., 2019a;b; Lee et al., 2020; Yarats et al., 2019), minimizing contrastive losses (Oord et al., 2018; Laskin et al., 2020b; Stooke et al., 2021; Zhang et al., 2022), clustering (Yarats et al., 2021b; Liu et al., 2023), and encouraging consistency under data augmentation (Laskin et al., 2020a; Yarats et al., 2021a). These approaches, however, may suffer when image observations contain complex distractor signals that are irrelevant to reward and/or control (Zhang et al., 2020; Fu et al., 2021; Wang et al., 2022; Lamb et al., 2022). Fu et al. (2021) and Wang et al. (2022) propose to explicitly model distractor signals and factor them out for control. Another idea is to learn representations through one-step (Pathak et al., 2017; Badia et al., 2020; Baker et al., 2022) or multi-step (Efroni et al., 2021; Lamb et al., 2022) action prediction. Although these approaches reduce irrelevant information, they do not explicitly induce structure in the representation space which directly helps policy learning (Zhang et al., 2020). Jaderberg et al. (2016) learn representation through many pseudo-reward functions, though the relevance of pseudo-rewards may be domain specific. Our work follows more closely from the line of work on learning representations based on behavioral distances (Ferns et al., 2004; Ferns & Precup, 2014), which we summarize below.

Representation Learning based on Behavioral Distances Behavioral distances between two states upper bound how "behaviorally" different they are, either under optimal policies (Ferns et al., 2004; 2011) or a fixed policy (Castro, 2020). One well-established example is the bisimulation metric (Ferns et al., 2004; 2011), which encodes both short-term differences in immediate rewards, and long-term differences between future state distributions. Zhang et al. (2020) incorporate the bisimulation metric into a policy learning framework where the metric and a corresponding state-representation are jointly learned. Specifically, they propose to learn a low-dimensional representation such that the L_1 distance in the representation space matches a learned approximation of the bisimulation metric. The induced metric structure facilitates reinforcement learning since the bisimulation metric provides an upper bound on the difference in values, which fosters value-based state-aggregation. New behavioral distances have since been proposed to simplify computation (Castro et al., 2021), improve training stability (Kemertas & Aumentado-Armstrong, 2021; Kemertas & Jepson, 2022), and reduce distance approximation errors (Chen & Pan, 2022). Castro et al. (2023) introduce a kernel interpretation of behavioral distances, yielding a new avenue for theoretical analysis. We discuss and compare the technical details of these approaches in later sections.

A crucial limitation of these existing approaches is that the behavioral distance to be modeled may not be realizable by the distance in the low-dimensional representation space (Castro et al., 2023). When combined with bootstrapping and function approximators, this realizability issue may undermine the quality of the learned representation and induce instability in training. In light of this issue, we propose focusing on preserving *local* metric structure, while allowing long-range distortion. We empirically show that representations learned to preserve local metric structure outperform existing approaches which try to preserve global structure (Zhang et al., 2020; Castro et al., 2023; Kemertas & Aumentado-Armstrong, 2021; Chen & Pan, 2022).

[1]We use top/bottom eigenvectors to refer to those corresponding to the largest/smallest non-zero eigenvalues respectively.

Spectral Methods in RL As suggested by the name, BeigeMaps rely on eigen-decompositions, and fall within the category of spectral RL methods. Most applications of spectral methods in RL can be traced back to proto value functions (PVFs) (Mahadevan & Maggioni, 2007), *reward-agnostic* eigenfunctions of Laplacians corresponding to transition dynamics graphs. Subsequent works have expanded on these ideas through the use of Krylov subspace methods (Petrik, 2007; Ghosh & Bellemare, 2020), singular value decompositions (Duan et al., 2019), spectral graph drawing (Wu et al., 2019; Wang et al., 2021), action-aware representations (Ren et al., 2023), and so on (Ghosh & Bellemare, 2020). PVFs are also known to correspond to the (scaled) eigenvectors of successor representations (Dayan, 1993; Stachenfeld et al., 2014). The EigenOptions framework (Machado et al., 2017a;b) builds upon this connection for option discovery and representation learning. As reward agnostic representation learning methods, these approaches attempt to capture the geometry of the transition dynamics and improve generalization over arbitrary reward functions.

Our proposed BeigeMaps are *reward-aware* spectral representations, and are most closely aligned with Comanici & Precup (2011), wherein Laplacian eigenmaps with respect to the bisimulation metric are used as state representations. While BeigeMaps also build upon similar ideas, our proposal extends the approach in Comanici & Precup (2011) along multiple directions. Namely, Comanici & Precup (2011) is restricted to tabular settings with enumerable states, with the additional constraint that both the policy and the behavioral metric are known a priori. Instead, we propose estimating behavioral eigenmaps for continuous state spaces in an end-to-end fashion where both the behavioral distance and the policy are simultaneously *learned* from data. These extensions necessitate various changes to the framework in Comanici & Precup (2011), including the adoption of *on-policy* behavioral distances (Castro, 2020), as well as neural approximations of eigenmaps (Deng et al., 2022b) to enable generalization in continuous state spaces.

3. Behavioral Distance Learning

We start by defining notation and the problem setting, which we formalize as a Markov decision process (MDP) defined by the tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{X}$ and $\mathcal{A}$ are continuous state and action spaces with dimensions $\texttt{dim}(\mathcal{X})$ and $\texttt{dim}(\mathcal{A})$, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The distribution $\mathcal{P}^a_x = \mathcal{P}(x'|x, a)$ encodes the probability of transitioning from state x to x' under action a. The RL objective is to train a policy mapping states to a distribution over actions, that maximizes the expected discounted return $\mathbb{E}_\pi\big[\sum_{t=0}^{H} \gamma^t r^a_{x_t}\big]$, where $r^a_x = \mathcal{R}(x, a)$, and H is the (potentially infinite) time horizon. The expectation $\mathbb{E}_\pi$ is taken over $\mathcal{P}^\pi_x$, the transition distribution under the policy π. A policy's *value* $V^\pi(x) = \mathbb{E}_\pi\big[\sum_{t=0}^{H} \gamma^t r^a_{x_t} | x_0 = x\big]$ is the expected return of following π when starting at x.

Behavioral distances define a notion of state dissimilarity that encodes differences in returns under a sequence of actions. They are termed “behavioral” precisely because they capture long-range temporal differences. These distances use a particular notion of dissimilarity, the bisimulation metric (Ferns et al., 2004; 2011), that forms an upper bound over state-value differences. This enables learning representations that group states by their values, which can be useful for downstream policy learning.

The Bisimulation (Pseudo) Metric The bisimulation metric $d^\sim(x, y)$ between two states $x, y \in \mathcal{X}$ is small if they yield similar immediate rewards, and transition to similar states (Ferns et al., 2004; 2011). It is defined as the unique fixed point of the following recursion

$$d^\sim(x, y) = \max_{a \in \mathcal{A}} \big[\, |r^a_x - r^a_y| + \gamma \mathcal{W}_1(\mathcal{P}^a_x, \mathcal{P}^a_y \,|d^\sim) \,\big].$$

Here, the 1-Wasserstein distance $\mathcal{W}_1(\cdot|d^\sim)$ *lifts* the ground distance $d^\sim$ (defined on $\mathcal{X}$) onto a distance between probability distributions (Peyré & Cuturi, 2018). Despite its name, $d^\sim$ is actually a *pseudo*-metric since non-identical states can still have zero distance (Ferns et al., 2011).

Having to enumerate over $\mathcal{X}$ makes $d^\sim$ intractable for continuous state spaces. Moreover, the maximization over actions makes it overly pessimistic – large distances may arise due to actions rarely executed under a policy (Castro, 2020). The *on-policy* bisimulation metric d^π resolves these issues by swapping the maximization over actions with expectations over *policy-induced* transitions (Castro, 2020). It is the unique fixed point of the following iteration:

$$d^\pi(x, y) = |\mathbb{E}_\pi(r_x) - \mathbb{E}_\pi(r_y)| + \gamma \mathcal{W}_1(\mathcal{P}^\pi_x, \mathcal{P}^\pi_y \,|d^\pi). \quad (1)$$

The distances $d^\sim$ and d^π define upper bounds on the differences in optimal values $|V^*(x) - V^*(y)| \le d^\sim(x, y)$ and the given policy's values $|V^\pi(x) - V^\pi(y)| \le d^\pi(x, y)$ respectively (Ferns et al., 2011; Castro, 2020). It is this connection to values that makes bisimulation metrics particularly useful for representation learning. If states are embedded in a space where their distances match d^π or $d^\sim$, similarly valued states should be grouped together.

Generalized Behavioral Distances Various modifications to the on-policy d^π have been proposed to improve computation and sample-based estimation, resulting in new behavioral distances (Zhang et al., 2020; Castro et al., 2021; 2023; Kemertas & Aumentado-Armstrong, 2021; Chen & Pan, 2022) that still retain the same form in Equation 1:

$$d^\pi(x, y) = d^\pi_R(x, y) + \gamma d^\pi_T(\mathcal{P}^\pi_x, \mathcal{P}^\pi_y \mid d^\pi). \quad (2)$$

Here, the first component d^π_R measures immediate reward-differences between states, while d^π_T measures differences in terms of their transition distributions.

Behavioral Similarities Offering a kernelized perspective, Castro et al. (2023) swap distances in Equation 2 for kernel functions $k(x, y)$ that measure *similarities* between states x and y. Kernels are often defined with respect to a distance; for instance, the Gaussian kernel $k(x, y) = \exp(-\eta\|x - y\|_2^2)$ with bandwidth $\eta > 0$ uses the L_2 distance. Kernels can also be defined without direct reference to a distance, as is the case in Castro et al. (2023). Despite the changes, the similarity measure given by Castro et al. (2023) still retains the form as Equation 2: $k^\pi(x, y) = k_R^\pi(x, y) + \gamma k_T(\mathcal{P}_x^\pi, \mathcal{P}_y^\pi \mid k^\pi)$. Since much of the literature uses the distance perspective, we follow the same and discuss the kernel interpretation in Section 6.

Representation Learning In representation learning, we seek an embedding $\phi : \mathcal{X} \to \Phi$ onto a feature space $\Phi \subseteq \mathbb{R}^D$, ideally with $D \ll \texttt{dim}(\mathcal{X})$. Behavioral distance learning algorithms parameterize the distance function in Equation 2 with a function approximator ϕ, and train it such that distances in Φ approximate d^π in $\mathcal{X}$.

These algorithms follow a common framework that can be categorized along three axes. The first is the choice of the *representational distance function* d_ϕ, the distance in Φ. The second axis is the choice for the reward and transition distances d_R^π an d_T^π. At every iteration, we compute target distances d^* between state pairs, using $d_R^\pi(x, y)$ and $d_T^\pi(\mathcal{P}_x^\pi, \mathcal{P}_y^\pi | d_\phi)$. In this *target distance update* step, we bootstrap the current representational distance d_ϕ as the ground distance for d_T^π. Finally, the parameters of ϕ are updated in the third *embedding update* step to minimize an objective $\mathcal{L}^\phi$ measuring errors in approximating d^* with d_ϕ. In Algorithm 1, we summarize the basic framework of behavioral distance algorithms, with an additional policy update step to simultaneously train a policy π.

Algorithm 1 Behavioral Distance Representation Learning

```
Data: Replay buffer $\mathcal{D}$, learning rates $\alpha_d$ and $\alpha_\pi$, neural
      network $\phi$ with weights $W_\phi$
1  # Define parameterized representation distance
2  rep_dist$(x, y)$ : return $d_\phi(x, y)$
3  for $i = 1$ to $\cdots$ do
4      $\{(x, a_x, r_x, x'), (y, a_y, r_y, y')\} \sim \mathcal{D}_{\text{Buffer}}$
5      # Target distance update
6      $d^*(x, y) \leftarrow d_R^\pi(r_x, r_y) + \gamma d_T^\pi(\mathcal{P}_x^\pi, \mathcal{P}_y^\pi | \texttt{rep\_dist})$
7      # Embedding update via distance loss
8      $W_\phi \leftarrow W_\phi - \alpha_d \nabla_{W_\phi} \sum_{x,y} \mathcal{L}^\phi(d^*(x, y), d_\phi(x, y))$
9      # Additional embedding update via policy loss
10     $W_\phi \leftarrow W_\phi - \alpha_\pi \nabla_{W_\phi} \texttt{PolicyLoss}(\phi)$
```

The Isomap Objective While behavioral distance algorithms have experimented with different representational distances and target update steps, they all essentially share the same embedding update step. Namely, the loss $\mathcal{L}^\phi$ corresponds to finding an isometric mapping from the distance space $(\mathcal{X}, d^\pi)$ to (Φ, d_ϕ). Drawing connections to the dimensionality reduction algorithm Isomap (Tenenbaum et al., 2000), we refer to this as the *isomap* objective defined below as an expectation over the data $\mathcal{D}$ in a replay buffer:

$$\mathcal{L}_{\text{IsoMap}}^\phi(d^*, d_\phi) = \mathbb{E}_{x,y\sim\mathcal{D}}\left[\left(d^*(x, y) - d_\phi(x, y)\right)^2\right]. \quad (3)$$

The isomap objective, while simple and intuitive, is unfortunately not well-defined for arbitrary distances; there may not exist a D-dimensional space Φ such that all pairwise distances d^* can be mapped exactly onto its metric (Bourgain, 1985; Linial et al., 1995; Belkin & Niyogi, 2001). Recently, Castro et al. (2023) tackled this question of embeddability for the reduced MICO distance, and showed that an isometric embedding is possible with the squared Euclidean distance in $\mathbb{R}^D$ with $D \le \texttt{dim}(\mathcal{X})$. However, this upper-bound is much too loose for practical purposes where we require $D \ll \texttt{dim}(\mathcal{X})$. In this context, Castro et al. (2023) show that a lower dimensional isometric embedding with $D < \texttt{dim}(\mathcal{X})$ is possible if one accepts a certain degree of distortion in the original metric. This raises a natural question: if isometry is ill-defined without some distortion, is there a certain type of acceptable distortion that still preserves important metric structure in $\mathcal{X}$?

We hypothesize that a good trade-off is to allow distortion in long-range distances while preserving the *local* metric structure of $(\mathcal{X}, d^\pi)$. Besides $\mathcal{L}_{\text{IsoMap}}^\phi$ being potentially ill-defined, this locality preserving relaxation is motivated by the fact that behavioral distances form an *upper*-bound on value differences. While a small behavioral distance implies that the states have similar values, large behavioral distances do not necessarily imply dissimilar values; they could even have identical values. Moreover, locality preserving embeddings are known to emphasize natural clusters in data (Belkin & Niyogi, 2001), which can be useful for value-based state aggregation.

Our proposed Behavioral Eigenmap representation replaces $\mathcal{L}_{\text{IsoMap}}$ with a new objective $\mathcal{L}_{\text{BeigeMap}}$ that emphasizes locality preservation. This objective is minimized by the eigenfunctions of the Laplacian defined by a behavioral distance d^π, as described in the next section.

4. BeigeMaps: Behavioral Eigenmaps

We now discuss Laplacian eigenmaps, which emerge as solutions to a locality preservation objective, and also exhibit additional properties motivating their use as representations for RL. We then discuss the NeuralEF algorithm (Deng et al., 2022b) used to approximate eigenfunctions of kernels. The BeigeMap objective is the NeuralEF objective with respect to kernels derived from behavioral distances.

4.1. Laplacian Eigenmaps

A reasonable approach to locality preservation is to use an objective function with distance-mismatch penalties inversely weighted by the distances themselves. Equivalently, we can encode distances d into similarities via a kernel function $k^d(\cdot,\cdot)$ such that smaller distances correspond to larger kernel similarities. We can then use these kernel evaluations as the weights. If instead of sampling states from $\mathcal{D}$, we just had a fixed set of states $\{x_i\}_{i=1}^N$ and the corresponding kernel matrix $K_{ij} = k^d(x_i, x_j)$, we could define the following locality preserving objective (Belkin & Niyogi, 2001):

$$\arg\min_{\phi} \sum_{i,j=1}^{N} \|\phi(x_i) - \phi(x_j)\|_2^2 \, K_{ij}, \text{ s.t. } \Phi^T \mathbb{D} \Phi = \mathbb{I}, \quad (4)$$

where $\Phi \in \mathbb{R}^{N\times D}$ is a matrix with $\phi(x_i)^T$ stacked along the rows. Note that this is a slight abuse of notation since we use Φ to also refer to the representation space – here Φ is being used to denote a finite set of points in this space. The objective in Equation 4 is high when similar points (with respect to d in $\mathcal{X}$) are mapped far apart (with respect to the L_2-distance in Φ) – promoting local neighborhoods in $\mathcal{X}$ to remain intact in Φ. The orthonormality constraint $\Phi^T\mathbb{D}\Phi = \mathbb{I}$ with respect to the *degree* matrix $\mathbb{D}_{ii} = \sum_{j=1}^N K_{ij}$, prevents the trivial solution where all embeddings are identical.

The optimal embeddings for Equation 4, known as Laplacian eigenmaps, are widely used for non-linear dimensionality reduction (Belkin & Niyogi, 2001). Given a high-dimensional dataset that lies on a latent lower-dimensional manifold, Laplacian eigenmaps provide a compact representation that captures the geometric structure of the underlying manifold. These eigenmaps can be computed exactly as the D bottom eigenvectors of the normalized Laplacian matrix $L = \mathbb{I} - \mathbb{D}^{-1}K$ (Belkin & Niyogi, 2001; Von Luxburg, 2007). Specifically, if we stack the D N-dimensional bottom eigenvectors of L into a matrix Φ, the i-th row of Φ is the optimal embedding for state x_i. Since the *top* eigenvectors of the *normalized kernel matrix* $\mathbb{D}^{-1}K$ correspond to the *bottom* eigenvectors of L, we can equivalently find the normalized kernel's top eigenvectors; we adopt this kernel based approach.

Connections to Spectral Clustering Laplacian eigenmaps are also used as embeddings in spectral clustering (Shi & Malik, 1997; Ng et al., 2001). Here, Laplacian eigenmaps embed the data in a feature space Φ that better reveals cluster structure, so that a simple clustering algorithm can then be applied on these features. This is motivated by the fact that Laplacian eigenvectors also solve the relaxed normalized cut problem of partitioning a graph into balanced clusters (Von Luxburg, 2007). Different clustering approaches may use different normalizations of the Laplacian, but their eigenmaps retain the same locality preserving property identified earlier (Von Luxburg, 2007). Since a key motivation for behavioral distances is to reveal value-based clusters in the state space, this connection to spectral clustering provides further motivation in using Laplacian eigenmaps as behavioral distance based representations.

Preventing Explosions and Collapses We defined the Laplacian eigenmap objective by relaxing the strict isometric objective. However, the orthonormality condition in Equation 4 introduces new constraints as well. So one may ask if these constraints introduce unwanted biases. Interestingly, they actually *resolve* known issues identified in existing behavioral distance algorithms.

Specifically, Kemertas & Aumentado-Armstrong (2021) find that some existing behavioral distance algorithms are prone to feature map *explosion*, where the norm of the embeddings grows beyond a required bound, and feature map *collapse*, where all embeddings collapse onto a single point in Φ. With Laplacian eigenmaps, the normalization constraint ensures that $\mathbb{E}(\|\phi\|_2^2) = D$, which prevents feature map explosion. On the other hand, the orthogonality constraint ensures that $\mathtt{dim}(\Phi) = D$, preventing collapse to a sub-space with dimension less than D. These are well-known properties of Laplacian eigenmaps, and are verified in Appendix A.1. Thus, the new constraints introduced by Laplacian eigenmaps are actually beneficial for behavioral distance based representation learning.

While matrix decompositions solve Equation 4 in closed form, these methods scale poorly with N and D, and do not generalize to unseen states. While Equation 4 assumes fixed data points, in practice, we need to generalize representations from a batch of samples from $\mathcal{D}$ to arbitrary states. We now describe the NeuralEF algorithm (Deng et al., 2022b), which approximates kernel eigenmaps through neural networks, enabling out-of-sample generalization.

4.2. Neural Eigenfunctions

Moving beyond matrix decompositions, we now generalize the notion of eigenvectors of kernel matrices to eigenfunctions of kernel functions. Given a kernel function $k(x, y)$, its eigenfunctions ϕ are characterized by the eigenvalue equation $\int_{\mathcal{X}} k(x, x')\phi(x')p(x')dx' = \mu\phi(x)$, for some measure $p(x)$ and eigenvalues μ. The Neural Eigenfunctions (NeuralEF) algorithm (Deng et al., 2022a; Deng & Luo, 2023) approximates the top eigenfunctions of k through neural networks by optimizing a differentiable objective function, the specifics of which we present in the next section.

For now, we note that with the use of NeuralEFs, we can approximate Laplacian eigenmaps through neural networks that can generalize and scale to infinite dimensional state spaces. Note that the NeuralEF algorithm provides the *top* eigenfunctions of the kernel, which correspond to the *bottom* eigenfunctions of the corresponding Laplacian, as required in Equation 4. Compared to other existing eigenfunction approximation algorithms (Pfau et al., 2018; Shaham et al., 2018), we found NeuralEFs to be particularly useful since they avoid expensive and potentially unstable matrix decompositions. A qualitative comparison with alternative methods in provided in Appendix A.2. We now describe our approach of learning behavioral distance based representations, trained using the NeuralEF algorithm.

4.3. BeigeMaps

We propose Behavioral Eigenmaps (BeigeMaps) as an alternative behavioral distance based representation for RL, and describe them in terms of the three axes defined in Section 3.

Representational Distance We use the L_2 distance in Φ as the representational distance for BeigeMaps, i.e., $d_\phi(x, y) = \|\phi(x) - \phi(y)\|_2$. This is because Equation 4 encodes (the inverse of) kernel similarities $k^d(x, y)$ into the L_2 norm in Φ – states close together in $\mathcal{X}$ with respect to d are mapped close in Φ with respect to the L_2 distance.

Target Distance Update When computing target distances d^*, we simply kernelize a base behavioral distance. Given a batch of state pairs $\{(x_i, y_i)\}$ and their corresponding target distances $d^*(x, y)$, we compute the batch kernel matrix $K^{d^*}(x, y) = \exp(-\eta d^*(x, y)^2)$, where $\eta > 0$ is the kernel's bandwidth. While we could choose alternate kernels as well, the Gaussian kernel is geometrically motivated as it approximates the heat kernel when d^* is viewed as the geodesic distance over a manifold (Varadhan, 1967).

We normalize the kernel K^{d^*} via the batch degree matrix $\mathbb{D}$. As discussed earlier, there are multiple choices for how the kernel matrix is normalized in the dimensionality reduction and spectral clustering literature. We consider two of the most common variants of the normalized kernel: $K_{sym} = \mathbb{D}^{-1/2} K \mathbb{D}^{-1/2}$ (symmetric) and $K_{rw} = \mathbb{D}^{-1} K$ (random-walk), following the nomenclature in Von Luxburg (2007). In our experiments, models with K_{sym} outperformed those with K_{rw}, so we report results using K_{sym}. We present comparisons between the two variants in Appendix A.7.

Embedding Update Finally, we compute BeigeMap representations by optimizing the following objective, corresponding to the NeuralEF loss (Deng et al., 2022b) applied to k^{d^*}, the target kernel function:

$$\phi_{\text{BeigeMap}} = \arg\min_{\phi} \mathcal{L}^{\phi}_{\text{BeigeMap}}$$
$$= \arg\min_{\phi} \frac{1}{B} \left(\beta \sum_{j=1}^{d} \sum_{i=1}^{j-1} \hat{C}_{ij}^2 - \sum_{j=1}^{d} C_{jj} \right),$$
$$\text{s.t. } \mathbb{E}[\phi \odot \phi] = \vec{1}\,,\ C = \Phi \hat{K}^{d^*} \Phi^T\,,\ \hat{C} = \hat{\Phi} \hat{K}^{d^*} \hat{\Phi}^T. \quad (5)$$

Here, β is a positive coefficient, $\odot$ is the Hadamard product (elementwise-multiplication), $\vec{1}$ is the all-ones vector, and $\hat{\mathbf{A}}$ corresponds to a matrix $\mathbf{A}$ computed with stop-gradients. The batch kernel matrix $K^{d^*} \in \mathbb{R}^{B \times B}$ consists of pair-wise evaluations of $k^{d^*}(x, y)$ over a batch of B states, and $\Phi \in \mathbb{R}^{B \times D}$ are their features computed via ϕ. The orthogonality constraint in Equation 4 has been slacked as a penalty in the first term scaled by β. To enforce the normalization constraint $\mathbb{E}[\phi \odot \phi] = \vec{1}$, we follow Deng et al. (2022b) and apply L_2 batch normalization after the last layer of ϕ. Note that while the NeuralEF algorithm does assume a positive definite kernel, it can also recover the top eigenfunctions of indefinite kernels as long as it has at least $D - 1$ positive eigenfunctions (Deng et al., 2022a).

In summary, the BeigeMap objective is the NeuralEF loss with respect to K^{d^*}, which is, in turn, defined with respect to the target behavioral distance d^*. The representations ϕ_{BeigeMap} are approximations of locality-preserving Laplacian eigenmaps (Equation 4) with respect to d^*. We have thus proposed BeigeMaps as a locality-preserving drop-in replacement for the isometric embedding objective in behavioral distance algorithms. As such, BeigeMaps complement these algorithms by providing an alternate means of learning representations from existing behavioral distances. We now discuss how BeigeMaps can be incorporated into various behavioral distances proposed in the literature.

5. Algorithms

Behavioral distance based RL algorithms jointly learn a policy π along with a representation ϕ induced by learned distances d^π. As shown in Algorithm 1, training repeatedly iterates over 1) representation updates with respect to the behavioral distances, 2) policy updates from the underlying policy learning algorithm, and 3) updates to any additional components such as a learned dynamics model or a reward prediction model. While behavioral distance learning is agnostic to the underlying policy learning method, much of prior work uses the Soft Actor Critic (SAC) algorithm (Haarnoja et al., 2018). In Figure 1, we illustrate their general framework, where blue (dark) boxes correspond to vanilla SAC components, and dashed boxes indicate various auxilliary components. The distance based representation learning components for BeigeMap algorithms are shown in yellow (light). The corresponding illustration for the baseline algorithms without BeigeMaps would simply remove the $K^{d^*}_{sym}$ component, and replace $\mathcal{L}^{\phi}_{\text{BeigeMap}}$ with $\mathcal{L}^{\phi}_{\text{IsoMap}}$.

We now discuss behavioral distance algorithms in the literature, and their BeigeMap variants. Table 1 summarizes the representational distance d_ϕ and target distance updates for all algorithms. We also present normalized returns averaged (interquartile median) across 7 DeepMind control environments in Figure 2 for reference. Experimental details and further analysis are provided in the next section.

Table 1: **Behavioral Distances** The three components of behavioral distance (top) and similarity (bottom) based algorithms.

Algorithm	Distance Update $d^* = d_R^\pi + \gamma d_T^\pi$	Representational Distance $d_\phi(x,y)$	Loss: $\mathcal{L}^\phi$
DBC	$\lvert r_x - r_y\rvert + \gamma\left(\lVert\mu_{\phi(x)} - \mu_{\phi(y)}\rVert_2^2 + \lVert\Sigma_{\phi(x)}^{1/2} - \Sigma_{\phi(y)}^{1/2}\rVert_F^2\right)$	$\lVert\phi(x) - \phi(y)\rVert_1$	$\mathcal{L}_{\text{IsoMap}}^\phi$
RobustDBC	$\lvert r_x - r_y\rvert + \gamma\left(\lVert\mu_{\phi(x)} - \mu_{\phi(y)}\rVert_2^2 + \lVert\Sigma_{\phi(x)}^{1/2} - \Sigma_{\phi(y)}^{1/2}\rVert_F^2\right)$	$\lVert\phi(x) - \phi(y)\rVert_2$	$\mathcal{L}_{\text{IsoMap}}^\phi$
MICO	$\lvert r_x - r_y\rvert + \gamma d_\phi(x', y')$	$\frac{1}{2}\lVert\phi(x)\rVert_2^2 + \frac{1}{2}\lVert\phi(y)\rVert_2^2 + \beta\theta(\phi(x), \phi(y))$	$\mathcal{L}_{\text{IsoMap}}^\phi$
RAP	$\sqrt{\lvert r_x - r_y\rvert^2 - \sigma_r^2(x) - \sigma_r^2(y)} + \gamma d_\phi(\mu_{\phi(x)}, \mu_{\phi(y)})$	$\frac{1}{2}\lVert\phi(x)\rVert_2^2 + \frac{1}{2}\lVert\phi(y)\rVert_2^2 + \beta\theta(\phi(x), \phi(y))$	$\mathcal{L}_{\text{IsoMap}}^\phi$
X+BeigeMap	Same as algorithm X	$\lVert\phi(x) - \phi(y)\rVert_2$	$\mathcal{L}_{\text{BeigeMap}}^\phi$

Algorithm	Kernel Update $k^* = k_R^\pi + k_T^\pi$	Representational Kernel $k_\phi(x,y)$	Loss: $\mathcal{L}^\phi$
KSME	$1 - \frac{r_x - r_y}{r_{\max} - r_{\min}} + \gamma k_\phi(x,y)$	$\phi(x)^T\phi(y)$	$\mathbb{E}([k^*(x,y) - k_\phi(x,y)]^2)$
KSME+Beigemap	Same as KSME	$\exp\left(-\eta\lVert\phi(x) - \phi(y)\rVert_2^2\right)$	$\mathcal{L}_{\text{BeigeMap}}^\phi$

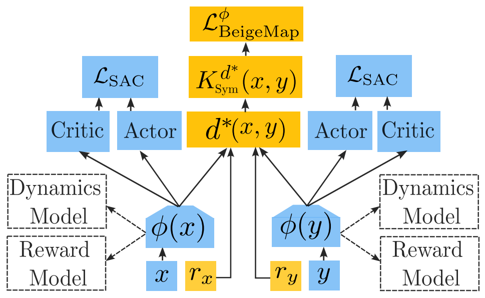

Figure 1: **BeigeMap Model Components** Blue boxes are components of the underlying SAC model, dashed boxes are auxiliary components, and yellow boxes are BeigeMap components.

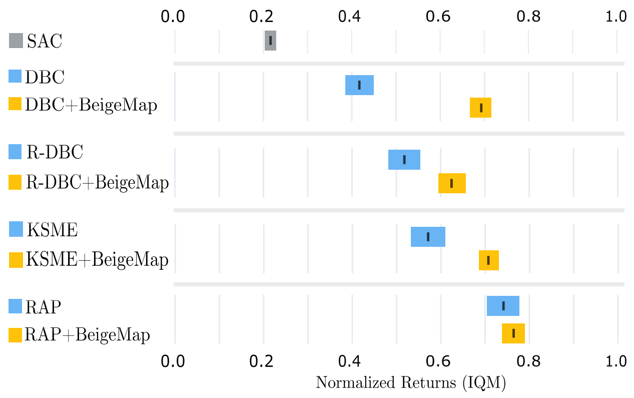

Figure 2: **Aggregate Model Performance** Interquartile median of normalized returns averaged over all seeds and environments. Shaded regions denote 95%-CI.

DBC The Deep Bisimulation for Control (DBC) algorithm was proposed as a gradient-based method to jointly learn a policy with a behavioral distance based representation for non-deterministic MDPs (Zhang et al., 2020). To compute the distance targets d^*, DBC sets the reward-based distance d_R^π to the absolute difference in rewards, and the transition distance d_T^π to the 2-Wasserstein distance $\mathcal{W}_2$ as an approximation of $\mathcal{W}_1$ in Equation 1. DBC also trains an approximate dynamics model parameterized as a Gaussian $\mathcal{N}(\mu_{\phi(x)}, \Sigma_{\phi(x)})$, with mean $\mu_{\phi(x)}$ and covariance $\Sigma_{\phi(x)}$, as well as a reward model predicting rewards from the next latent states. Using the approximate Gaussian dynamics model, the $\mathcal{W}_2$ distances are calculated in closed-form as the L_2 distance between the means μ_ϕ and the Frobenius norm distance between the square-root of covariances $\Sigma_{\phi(x)}^{1/2}$ as shown in Table 1. While simplifying computation, this approximation introduces a mismatch between the representational distance d_ϕ (using L_1) and the ground distance for d_T^π (using L_2) (Kemertas & Aumentado-Armstrong, 2021).

R-DBC The Robust DBC (R-DBC) (Kemertas & Aumentado-Armstrong, 2021) algorithm corrects some training instabilities that can arise with DBC. Specifically, DBC training can diverge due to embedding explosion and embedding collapse. Embedding explosions are characterized by embedding norms growing beyond the threshold[2] $\frac{r_{\max} - r_{\min}}{2(1-\gamma)}$, a stability bound derived by the authors. Embedding collapse, on the other hand, can occur for environments with sparse or near-constant rewards. In such environments, the representation ϕ can prematurely collapse onto a single point $\phi(x) = \phi_0 \forall x \in \mathcal{X}$ since the model initially observes all states to essentially have zero behavioral distance (Kemertas & Aumentado-Armstrong, 2021).

While retaining the behavioral distance in DBC (but fixing distances to L_2), R-DBC proposes a few algorithmic changes. To prevent embedding explosion, R-DBC clips the embedding norms so they remain under the required threshold. To prevent embedding collapse, R-DBC learns an additional inverse dynamics model in the latent space predicting actions from latent state transitions, and adds a curiosity-driven intrinsic reward to the original rewards. This intrinsic reward is set proportional to errors between the predicted next latent state from the learned dynamics model

[2]Kemertas & Aumentado-Armstrong (2021) use a generalized version of the bisimulation metric leading to the bound $\frac{c_R(r_{\max} - r_{\min})}{2(1-c_T)}$ for $c_T \in [0,1), c_R \in [0,\infty)$. For d^π in Equation 1, this corresponds to setting $c_T = \gamma, c_R = 1$.

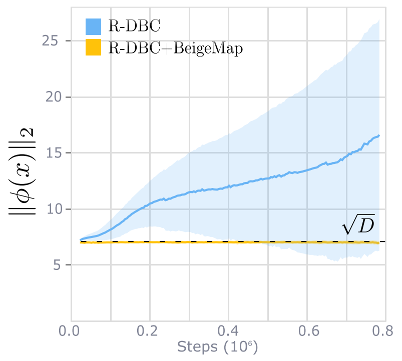

Figure 3: **Preventing Embedding Explosion** Mean norm (and shaded standard deviation) of embeddings during training, averaged over all environments. The dashed line indicates $\sqrt{D} = \sqrt{50} \approx 7.1$.

and the actual next latent state, encouraging exploration to states where dynamics modeling needs improvement.

In Figure 2, we see that the proposed changes do indeed improve performance over DBC. But we note that both R-DBC+BeigeMaps and, interestingly, DBC+BeigeMaps outperform R-DBC. This is likely because embedding explosion and collapse, the two issues that R-DBC solves, are both inherently mitigated by BeigeMap's orthonormality constraints as discussed in Section 4. Specifically, the normalization constraint encourages $\|\phi\|_2^2 \sim D$, and the orthogonality penalty encourages $\mathtt{dim}(\Phi) \sim D$. In Figure 3, we see that $\|\phi\|_2$ converges to $\sqrt{D}$ for R-DBC+BeigeMap, while it continues to grow for R-DBC.

KSME Castro et al. (2023) take a kernelized approach to behavioral distances, defining state *similarilaties* through kernel functions as opposed to differences. However, KSME still follows the same framework we have been using with certain kernel based modifications. Instead of representational distances d_ϕ, KSME defines representational similarities encoded as the inner-product $k_\phi(x, y) = \phi(x)^T\phi(y)$. The target distance update step is then replaced by a target kernel update step $k^*(x, y) = k_R^\pi(x, y) + \gamma k_T(\mathcal{P}_x^\pi, \mathcal{P}_y^\pi \mid k_\phi)$. Corresponding to the reward differences d_R^π in other methods, KSME uses the similarity kernel $k_R^\pi(x, y) = 1 - \frac{r_x - r_y}{r_{\text{max}} - r_{\text{min}}}$. The transition-based similarities are encoded in the *lifted* kernel $k_T(\mathcal{P}_x^\pi, \mathcal{P}_y^\pi | k_\phi) = \mathbb{E}(k_\phi(x, y))$, where the expectation is taken over $x' \sim \mathcal{P}_x^\pi$ and $y \sim \mathcal{P}_y^\pi$. Finally, KSME optimizes the kernel equivalent of the isomap objective where we try to match all pair wise kernel similarities.

Castro et al. (2023) demonstrate that KSME induces a behavioral distance equivalent to the reduced MICO (Matching under Independent Couplings) distance, which approximates the $\mathcal{W}_2$ under independent couplings of distributions (Castro et al., 2021). Moreover, the policies trained using KSME were shown to be empirically equivalent to those trained using MICO. Thus, we use KSME as a substitute for MICO in our experiments.

To define the BeigeMap variant of KSME, we set the representational kernel $k_\phi(x, y)$ to be the Gaussian kernel as shown in Table 1. This choice maintains consistency with other BeigeMap algorithms, and is also geometrically motivated for its connections to the heat kernel. We also replace the kernel matching objective with the NeuralEF objective defined over the normalized kernel obtained from KSME's target kernel updates.

RAP The Reducing Approximation Gap (RAP) distance attempts to mitigate various approximation errors in existing behavioral distance algorithms (Chen & Pan, 2022). Firstly, RAP replaces the $\mathcal{W}_2$ distance in DBC with a sample based divergence $\mathbb{E}_{a_x\sim\pi, a_y\sim\pi}(d_\phi(\mathbb{E}(x'), \mathbb{E}(y')))$, where $\mathbb{E}(x') = \mathbb{E}_{x'\sim P_x^{a_x}}$ is an expectation over next states. To estimate this distributional distance, RAP trains an approximate Gaussian dynamics model with mean $\mu_{\phi(x)}$, and sets $\mathbb{E}(x') = \mu_{\phi(x)}$. The representational distance d_ϕ is $\frac{1}{2}\|\phi(x)\|_2^2 + \frac{1}{2}\|\phi(y)\|_2^2 + \beta\theta(\phi(x), \phi(y))$, where $\theta(\cdot)$ is the angle between vectors.

Secondly, RAP approximates the reward-based distance when computing the target d^* as $d_R^\pi(x, y) = |\mathbb{E}_{a_x\sim\pi} r_x^{a_x} - \mathbb{E}_{a_y\sim\pi} r_y^{a_y}|$, whereas existing algorithms simply use $|r_x^{a_x} - r_y^{a_y}|$. This is estimated through the covariances σ_r of a learned Gaussian reward model model $\mathcal{N}(\mu_r, \sigma_r)$ as shown in Table 1. Finally, RAP uses a slightly modified objective $\mathcal{L}_{\text{RAP}} = \mathbb{E}_{\mathcal{D}}[(d_\phi - \gamma d_S^\pi(x, y))^2 - d_S^{\pi\,2}]$ to avoid square roots. Despite this algebriac change, the new objective still corresponds to finding an isometric mapping ϕ.

6. Experiments

To evaluate the effectiveness of our proposal, we augment the algorithms in Section 5 with BeigeMaps and demonstrate improvements in policy learning over all baseline behavioral distance algorithms.

DM Control Environment We train and evaluate all algorithms on the DeepMind Control (DMC) suite (Tassa et al., 2018), a set of continuous control tasks that has been used as a benchmark for all prior behavioral distance algorithms. We picked the following 7 environments from the suite to maximize overlap with prior work: `finger_spin`, `walker_stand`, `walker_walk`, `cheetah_run`, `cartpole_swingup`, `cartpole_balance`, and `ball_in_cup_catch`. We use RGB pixel observations of size $(3, 88, 88)$, and stack 3 consecutive frames to account for partial observability. All environments are truncated at 1000 steps and have dense rewards bounded between $[0, 1]$, except for `ball_in_cup_catch` which has sparse binary rewards.

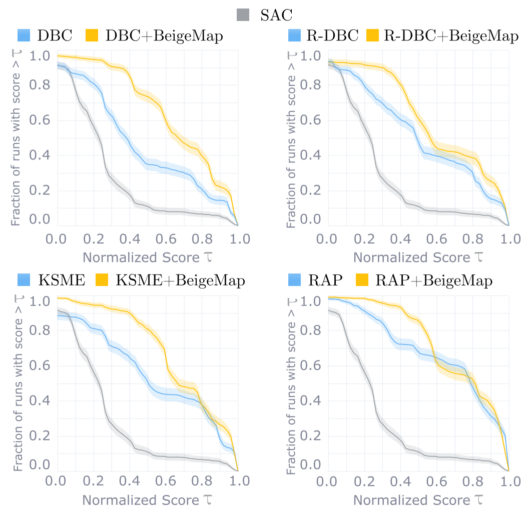

Figure 4: **Performance Profiles** Comparing algorithms with respect to multiple performance thresholds.

We normalize the default episodic returns within $[0, 1000]$ to $[0, 1]$ to standardize all results. Further details on environment parameters are provided in Appendix A.3.

Model Architectures We use Soft Actor Critic (SAC) as the underlying RL model for all algorithms (Figure 1). All algorithms also learn a Gaussian latent dynamics model, parameterized by an MLP. DBC, R-DBC, and RAP algorithms also learn a latent reward prediction model parameterized by an MLP. We fixed all hyperparameters to match those used by Zhang et al. (2020), which has been a common baseline for all other prior works. Due to computational constraints, we reduced the replay buffer size from $1,000,000$ to $100,000$. For BeigeMap algorithms, we set the kernel bandwidth using the median-heuristic (Garreau et al., 2017). We list all model hyperparameters and network architecture details in Appendix A.4.

6.1. Results

We compare the performance of all baseline and BeigeMap algorithms using multiple robust evaluation metrics recommended by Agarwal et al. (2021). For each algorithm, we used the model parameters that resulted in the highest return during training, and obtained normalized returns from the corresponding policies for 30 random evaluation seeds. We then repeat the process for 3 additional random training seeds, resulting in a total of 90 evaluations for each algorithm on every environment. For comparison, we also present results for the vanilla SAC algorithm. In all plots, the shaded error bars correspond to 95% confidence intervals obtained through stratified bootstraping over $10,000$ samples with replacement computed using the `rliable` library (Agarwal et al., 2021). We also present additional analyses in Appendix A.8

Interquartile Medians Figure 2 shows the interquartile median (IQM) normalized returns for all models averaged over all environments and random seeds. The BeigeMap variants (yellow) considerably outperform their respective baselines (blue) for DBC, R-DBC, and KSME. The RAP+BeigeMap variant does obtain higher average returns than RAP, but with some confidence-interval overlap. Finally, the variance between the different BeigeMap models is lower than that between the baselines. The worst performing BeigeMap model (R-DBC + BeigeMap) still outperforms an average baseline model.

Performance Profiles Point estimates such as the IQM may not be best suited to reveal performance variability across tasks (Agarwal et al., 2021). Thus, in Figure 4, we present performance profiles that tend to better capture these differences. For each model, we calculate the proportion of trials (across all environments and seeds) where the model's returns exceeded various thresholds τ. A model is said to *stochastically dominate* another if its curve is strictly above the latter (Agarwal et al., 2021). BeigeMap variants stochastically dominate their counter parts for DBC, R-DBC, and KSME. Moreover, the RAP+BeigeMap curve is strictly above RAP except for a small range of thresholds. This provides additional context to the IQM comparisons earlier, suggesting that BeigeMap+RAP is the better option for most performance thresholds.

7. Conclusion

We presented BeigeMap, a representation learning method that replaces the isometric objective in behavioral distance algorithms with a locality preserving objective that highlights natural clusters in the state space. BeigeMaps are eigenfunctions of kernels induced by behavioral distances, and can be incorporated as a drop-in replacement for the representational update step in existing behavioral distance algorithms. We demonstrated that adding BeigeMaps improved performance for all behavioral distances evaluated.

Limitations and Future Work Prior works have demonstrated benefits of behavioral distance based representations in terms of robustness to task-irrelevant distractors in observations (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Chen & Pan, 2022). In this work, we focused on demonstrating the effectiveness of BeigeMaps in RL from images in general. However, the formulation of BeigeMaps as Laplacian eigenfunctions can be used to define regularization operators that bias solutions towards functions that exhibit minimal variance within clusters (Smola & Kondor, 2003). Such regularization could help improve generalization of the learned representations with respect to distractors. While this was beyond the scope of our current work, it offers a fruitful direction for further research.

Acknowledgements

All experiments were done using the Hyak computing cluster at the University of Washington. We thank the anonymous reviewers at ICML 2024 for engaging in discussions and providing valuable recommendations.

Impact Statement

We present work aimed at advancing methods for reinforcement learning, a field with a broad range of applications in society and industry. In this work, we have attempted to provide algorithmic improvements, and have performed evaluations in simulation – presenting minimal direct societal impact or ethical concerns. However, as with any machine learning model, reinforcement learning algorithms should be deployed with care to prevent biases and harm.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pp. 507–517. PMLR, 2020.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Belkin, M. and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

Bourgain, J. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

Castro, P. S. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10069–10076, 2020.

Castro, P. S., Kastner, T., Panangaden, P., and Rowland, M. Mico: Improved representations via sampling-based state similarity for markov decision processes. In *Neural Information Processing Systems*, 2021.

Castro, P. S., Kastner, T., Panangaden, P., and Rowland, M. A kernel perspective on behavioural metrics for markov decision processes. *Transactions on Machine Learning Research*, 2023.

Chen, J. and Pan, S. Learning representations via a robust behavioral metric for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 36654–36666, 2022.

Comanici, G. and Precup, D. Basis function discovery using spectral clustering and bisimulation metrics. In *International Workshop on Adaptive and Learning Agents*, pp. 85–99. Springer, 2011.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Deng, Z. and Luo, Y. Learning neural eigenfunctions for unsupervised semantic segmentation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 551–561, 2023.

Deng, Z., Shi, J., Zhang, H., Cui, P., Lu, C., and Zhu, J. Neural eigenfunctions are structured representation learners. *ArXiv*, abs/2210.12637, 2022a.

Deng, Z., Shi, J., and Zhu, J. Neuralef: Deconstructing kernels by deep neural networks. In *International Conference on Machine Learning*, pp. 4976–4992. PMLR, 2022b.

Duan, Y., Ke, T., and Wang, M. State aggregation learning from markov transition data. *Advances in Neural Information Processing Systems*, 32, 2019.

Efroni, Y., Misra, D., Krishnamurthy, A., Agarwal, A., and Langford, J. Provably filtering exogenous distractors using multistep inverse dynamics. In *International Conference on Learning Representations*, 2021.

Ferns, N. and Precup, D. Bisimulation metrics are optimal value functions. In *UAI Conference on Uncertainty in Artificial Intelligence*, pp. 210–219, 2014.

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite markov decision processes. In *AAAI Conference on Artificial Intelligence*, 2004.

Ferns, N., Panangaden, P., and Precup, D. Bisimulation metrics for continuous markov decision processes. *SIAM J. Comput.*, 40:1662–1714, 2011.

Fu, X., Yang, G., Agrawal, P., and Jaakkola, T. Learning task informed abstractions. In *International Conference on Machine Learning*, pp. 3480–3491. PMLR, 2021.

Garreau, D., Jitkrittum, W., and Kanagawa, M. Large sample analysis of the median heuristic. *arXiv: Statistics Theory*, 2017.

Ghosh, D. and Bellemare, M. G. Representations for stable off-policy reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3556–3565. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/ghosh20b.html.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019a.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2016.

Kemertas, M. and Aumentado-Armstrong, T. Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34:4764–4777, 2021.

Kemertas, M. and Jepson, A. Approximate policy iteration with bisimulation metrics. *Transactions on Machine Learning Research*, 2022.

Lamb, A., Islam, R., Efroni, Y., Didolkar, A. R., Misra, D., Foster, D. J., Molu, L. P., Chari, R., Krishnamurthy, A., and Langford, J. Guaranteed discovery of control-endogenous latent states with multi-step inverse models. *Transactions on Machine Learning Research*, 2022.

Lange, S. and Riedmiller, M. Deep auto-encoder neural networks in reinforcement learning. In *The 2010 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2010.

Lange, S., Riedmiller, M., and Voigtländer, A. Autonomous reinforcement learning on raw visual input data in a real world application. In *The 2012 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2012.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020a.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020b.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

Linial, N., London, E., and Rabinovich, Y. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

Liu, Q., Zhou, Q., Yang, R., and Wang, J. Robust representation learning by clustering with bisimulation metrics for visual reinforcement learning with distractions. In *AAAI Conference on Artificial Intelligence*, 2023.

Machado, M. C., Bellemare, M. G., and Bowling, M. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pp. 2295–2304. PMLR, 2017a.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. Eigenoption discovery through the deep successor representation. *In ICLR International Conference on Learning Representations*, 2017b.

Mahadevan, S. and Maggioni, M. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(10), 2007.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Nyström, E. J. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. 1930.

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

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Petrik, M. An analysis of laplacian methods for value function approximation in mdps. In *IJCAI*, pp. 2574–2579, 2007.

Peyré, G. and Cuturi, M. Computational optimal transport. *Found. Trends Mach. Learn.*, 11:355–607, 2018.

Pfau, D., Petersen, S., Agarwal, A., Barrett, D. G. T., and Stachenfeld, K. L. Spectral inference networks: Unifying deep and spectral learning. In *International Conference on Learning Representations*, 2018.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007.

Ren, T., Zhang, T., Lee, L., Gonzalez, J. E., Schuurmans, D., and Dai, B. Spectral decomposition representation for reinforcement learning. *In ICLR International Conference on Learning Representations*, 2023.

Shaham, U., Stanton, K. P., Li, H., Nadler, B., Basri, R., and Kluger, Y. Spectralnet: Spectral clustering using deep neural networks. *In ICLR International Conference on Learning Representations*, 2018.

Shi, J. and Malik, J. Normalized cuts and image segmentation. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 731–737, 1997.

Smola, A. J. and Kondor, R. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003*, pp. 144–158. Springer, 2003.

Stachenfeld, K. L., Botvinick, M., and Gershman, S. J. Design principles of the hippocampal cognitive map. *Advances in neural information processing systems*, 27, 2014.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. P., and Riedmiller, M. A. Deepmind control suite. *ArXiv*, abs/1801.00690, 2018.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Varadhan, S. R. S. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics*, 20 (2):431–455, 1967.

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

Wang, K., Zhou, K., Zhang, Q., Shao, J., Hooi, B., and Feng, J. Towards better laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning*, pp. 11003–11012. PMLR, 2021.

Wang, T., Du, S., Torralba, A., Isola, P., Zhang, A., and Tian, Y. Denoised mdps: Learning world models better than the world itself. In *International Conference on Machine Learning*, pp. 22591–22612. PMLR, 2022.

Williams, C. K. I. and Seeger, M. W. Using the nyström method to speed up kernel machines. In *Neural Information Processing Systems*, 2000.

Wu, Y., Tucker, G., and Nachum, O. The laplacian in rl: Learning representations with efficient approximations. *In ICLR International Conference on Learning Representations*, 2019.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In *AAAI Conference on Artificial Intelligence*, 2019.

Yarats, D., Fergus, R., and Kostrikov, I. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021*, 2021a.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021b.

Yen-Chen, L., Zeng, A., Song, S., Isola, P., and Lin, T.-Y. Learning to see before learning to act: Visual pre-training for manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7286–7293. IEEE, 2020.

Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. *International Conference on Learning Representations*, 2020.

Zhang, T., Ren, T., Yang, M., Gonzalez, J., Schuurmans, D., and Dai, B. Making linear mdps practical via contrastive representation learning. In *International Conference on Machine Learning*, pp. 26447–26466. PMLR, 2022.

A. Appendix

A.1. BeigeMaps Prevent Feature Map Collapse and Explosion

Normalization Constraints Prevent Feature Map Explosion The normalization constraints of BeigeMaps constrain the expected value of the squared L_2 norm of the feature maps to be equal to the feature-dimension. Since BeigeMaps naturally enforce this constraint, additional heuristic clipping is not required.

This follows from standard properties of the normalization constraint of Laplacian eigenmaps. This constraint is enforced by the NeuralEF algorithm used in BeigeMaps as $\mathbb{E}(\vec{\phi} \odot \vec{\phi}) = \vec{1}$, where $\vec{\phi}_x$ is the learned eigenmap for state x, $\vec{1}$ is the all-ones vector, and $\odot$ is the element-wise product. Let us now consider the expected value of $\mathbb{E}(\|\phi(x)\|_2^2)$ for a batch of B embeddings $\phi(x) \in \mathbb{R}^D$

$$
\begin{aligned}
\mathbb{E}(\|\phi(x)\|_2^2) &= \frac{1}{B} \sum_x \|\phi(x)\|_2^2 \\
&= \frac{1}{B} \sum_x \sum_i \phi(x)[i]\phi(x)[i] \\
&= \sum_i^d \frac{1}{B} \sum_x \phi(x)\phi(x)[i] \\
&= \sum_i^d \left(\frac{1}{B} \sum_x \phi(x)\phi(x) \right) [i] \\
&= \sum_i^d \mathbb{E}(\phi(x) \odot \phi(x))[i] \\
&= \sum_i^d \vec{1}[i] \text{ from the constraint} \\
&= D
\end{aligned}
$$

Hence, $\mathbb{E}(\|\phi(x)\|_2^2) = D$. Empirically, this normalization constraint is enforced by the $L-2$ batch normalization layer applied after the last layer of the encoder ϕ. We also confirm the above result experimentally, as shown in Figure 3.

Orthogonality Constraints Prevent Feature Map Collapse Feature map collapse is prevented by the orthogonality constraint of Laplacian eigenmaps $\Phi^T \mathbb{D} \Phi = \mathbb{I}$, where $\Phi \in \mathbb{R}^{B \times D}$ is the matrix of feature maps for B states, where the i-th row is the feature map for the i-th state x_i. Here, $\mathbb{D}$ is the diagonal degree matrix consisting column sums of the kernel. Since $\mathbb{D}$ is a positive definite matrix, we can write the orthogonality constraint as

$$
\begin{aligned}
\Phi^T \mathbb{D}^{1/2} \mathbb{D}^{1/2} \Phi = \mathbb{I} \\
(\mathbb{D}^{1/2} \Phi)^T (\mathbb{D}^{1/2} \Phi) = \mathbb{I}
\end{aligned}
$$

Thus, the matrix $\mathbb{D}^{1/2}\Phi$ has full-column rank. This implies that Φ has rank D by using the property that $\texttt{rank}(AB) = \texttt{rank}(A)$ if C is an $B \times D$ matrix of rank D. Since the rank of Φ (i.e. the dimension of the sub-space spanned by the feature map) is constrained to be D, Laplacian eigenmaps naturally prevent feature map collapse. Similar arguments discussing the dimensionality of Φ can also be found in Belkin & Niyogi (2001). The NeuralEF algorithm used in BeigeMaps applies this orthogonlization constraint as a penalty in the overall objective.

A.2. Neural Eigenfunction Algorithms

NeuralEFs are not the only approach to learning eigenfunctions capable of out-of-sample generalization. Random Fourier Features (RFFs) (Rahimi & Recht, 2007) and the Nystrom method (Nyström, 1930; Williams & Seeger, 2000) are some classic approaches. RFFs are not ideal for non shift-invariant kernels and generally require feature maps with large dimensionality D. The Nystrom method relies on explicit matrix eigendecomposition, which can be prohibitively expensive for large N. More recent approaches include Spectral Inference Networks (SpIN) (Pfau et al., 2018) and SpectralNets (Shaham et al., 2018) . Both of these methods rely on Cholesky decompositions, which is computationally expensive and tends to introduce instabilities. We opted for NeuralEFs since they do not suffer from the drawbacks described here, as well as the fact the NeuralEFs are shown to be applicable to indefinite kernels as well (as long as the kernel has more than $D - 1$ positive eigenvalues). Theoretically, BeigeMaps could be learned with any of these or other eigenfunction learning approaches.

A.3. DeepMind Control Suite Environments

In Table 2, we provide the environment parameters for the DeepMind Control suite environments used in our experiments. The choice of action repeats for various environments has been noted as important, but can vary across implementations in the literature. We use the action repeats that were shown to work well in Yarats et al. (2019), which are also listed in Table 2.

Parameter	Value
Image Dimension	3 x 88 x 88
Stacked Frames	3
Observation Dimension	9 x 88 x 88
Episode Length	1000
Action Repeats	
Finger Spin	2
Walker Walk	2
Walker Stand	2
Cheetah Run	4
Cartpole Swingup	8
Cartpole Balance	8
Ball in Cup Catch	4

Table 2: Parameters used for DMC environments.

A.4. Model Hyperparameters

We use the same model architecture as in Zhang et al. (2020), and attempt to match their implementation and hyperparameters as closely as possible. In Table 3, we list all model hyperparameter choices – the only difference from Zhang et al. (2022) is that we use a smaller replay buffer size ($100,000$ instead of $1M$) due to computational constraints. To enforce the normalization constraints for BeigeMaps in Equation 5, we apply an L_2 normalization layer at the end of the encoder following Deng et al. (2022b).

A.5. Training Infrastructure

All models were trained using the Hyak computing cluster at the University of Washington. Each model was trained on a single GPU, which was assigned by the cluster's scheduling system. Due to the cluster's scheduling constraints, training jobs could be paused and passed to a different node at any time. To ensure consistency across all nodes, we used a Docker container to recreate the same system settings across all machines used during training. Using this computing infrastructure, each model took about $20-30$ hours to train for $1M$ steps, including potential downtime waiting for the cluster to reschedule jobs. We did not observe significant difference in computation time for Beigemap and Isomap models, which took an average of 0.0823 and 0.0824 seconds per training step respectively. Note that the training could potentially be sped up with parallelization of environment rollouts (e.g. multiple vectorized environments) and model updates. We opted not to parallelize for better comparison with prior work that did the same.

A.6. Learning Curves

In Figure 5, we present normalized returns obtained from all models during training. As in the main text, the returns (which are within $[0, 1000]$) are normalized to be within $[0, 1]$. During training, we recorded each algorithm's average returns over 3 evaluation environments with different seeds. We repeat this training for 3 random training seeds, and report the average over all seeds here. The shaded regions correspond to the standard deviation in these mean estimates.

Hyperparameter	Value
Replay buffer capacity	**100,000**
Batch Size	128
Discount γ	0.99
Critic learning rate	10^{-5}
Critic target update frequency	2
Critic Q-function soft-update rate τ_Q	0.005
Critic encoder soft-update rate τ_ϕ	0.005
Actor learning rate	10^{-5}
Actor update frequency	2
Actor log stddev bounds	[-5,2]
Decoder learning rate	10^{-5}
Temperature learning rate	10^{-4}
Temperature Adam's β_1	0.9
Init Temperature	0.1
Dynamics hidden dimension	512
Encoder	
Encoder feature dimension	50
Encoder learning rate	10^{-5}
Number of convolutional layers	4
Num filters per convolution	32
Kernel size	3 x 3
Stride	2 for first, 1 for rest
Actor	
Actor MLP num layers	2
Actor MLP hidden dimension	256
Critic	
Critic MLP layers	2
Critic MLP hidden dimension	256
Gaussian Dynamics Model	
MLP num layers	1
MLP hidden dimension	512
Reward Prediction Model	
MLP num layers	1
MLP hidden dimension	512

Table 3: Model Hyperparameters

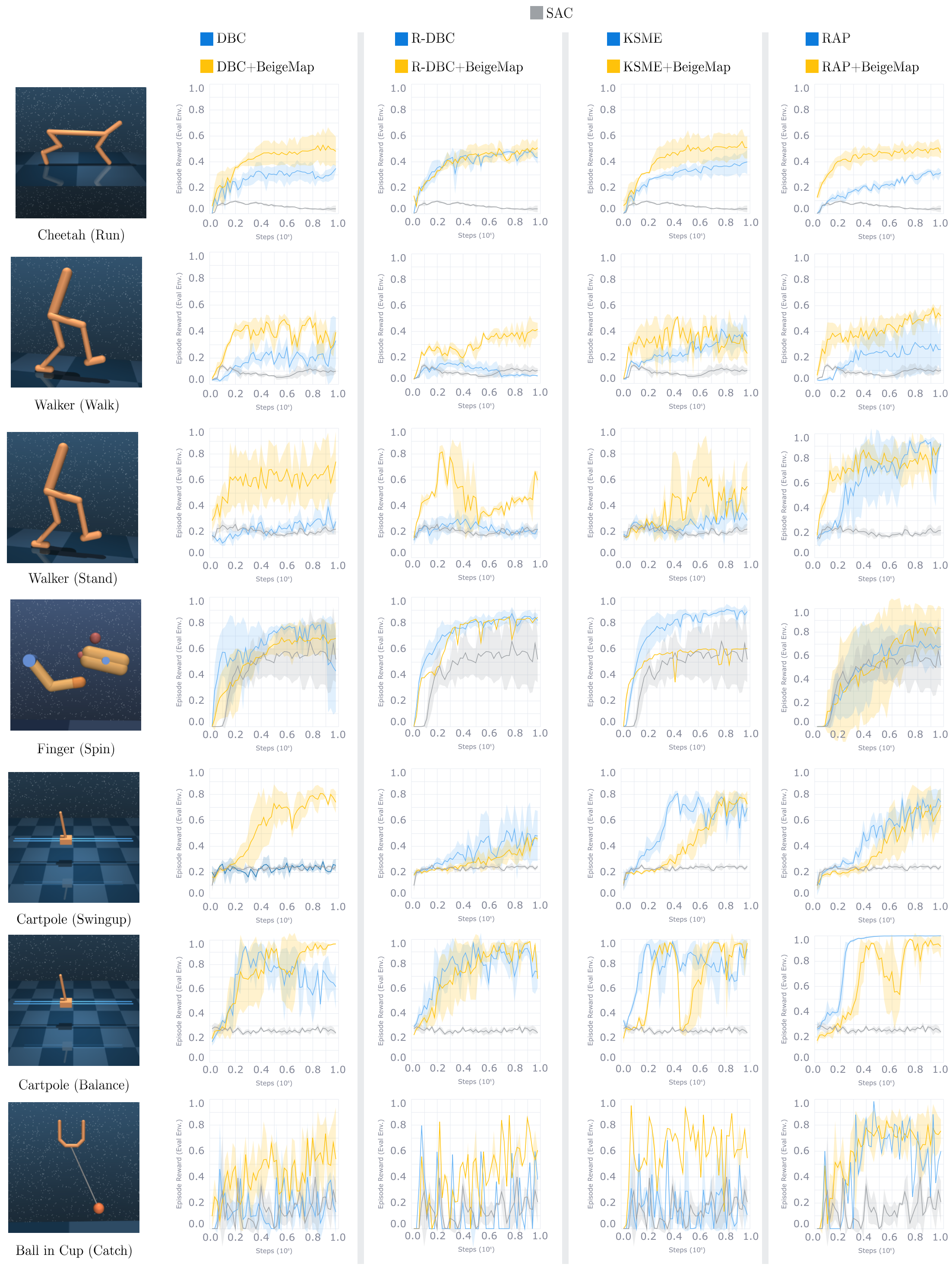

Figure 5: Learning curves for all environments. Mean normalized returns on evaluation environment averaged over 3 random seeds.

A.7. Kernel Normalization: Symmetric vs. Random Walk

As discussed in the main text, the literature on spectral clustering and dimensionality tend to use a few variations on how the Laplacian matrix (or in our case, the kernel matrix) is normalized. While different, all normalizations are intimately related, and share similar interpretations with respect to locality preservation and clustering. We consider two of the most popular alternatives: the symmetric normalization $K_{sym} = \mathbb{D}^{-1/2}K\mathbb{D}^{-1/2}$ and the random-walk normalization $K = \mathbb{D}^{-1}K$, following the nomenclature from Von Luxburg (2007). The relationship between these normalization choices is described in Proposition 3 in Von Luxburg (2007), which we summarize here.

The Laplacian eigenmaps corroesponding to the locality preserving objective in Equation 4 correspond to bottom solutions of the *generalized* eigenvalue problem $L\phi = \lambda\mathbb{D}\phi$ (Belkin & Niyogi, 2001). If (λ, ϕ) is an eigenvalue-eigenvector pair for the genralized eigenvalue problem, it is also an eigenpairs of the random-walk Laplacian $L_{rw} = \mathbb{D}^{-1}K$, i.e.$L_{rw}\phi = \lambda\phi$. Moreover, if (λ, ϕ) is an eigenpair of L_{rw}, then $(\lambda, D^{1/2}\phi)$ is an eigenpair of $L_{sym} = \mathbb{D}^{-1/2}K\mathbb{D}^{-1/2}$. Thus the eigenspectrum of L_{rw} and L_{sym} only differ by a factor of $\mathbb{D}^{1/2}$. Note that the bottom eigenvectors of either normalized Laplacian corresponds to the top eigenvector of the corresponding normalized Kernel. When computing BeigeMaps, we use the normalized kernel instead of the normalized Laplacian.

In Figure 6, we compare two options for normalized kernels used in BeigeMaps: the symmetric (Sym) normalization $K = \mathbb{D}^{-1/2}K\mathbb{D}^{-1/2}$ and $K = \mathbb{D}^{-1}K$. In the main text, we present results for the symmetric normalization as it generally outperformed the random-walk normalization for all distances evaluated. Note that while we conducted trials over multiple seeds for the experiments in the main text, the comparisons in Figure 6 only used a single seed. This was done to maximize the use of our limited computational resources for the main experiments.

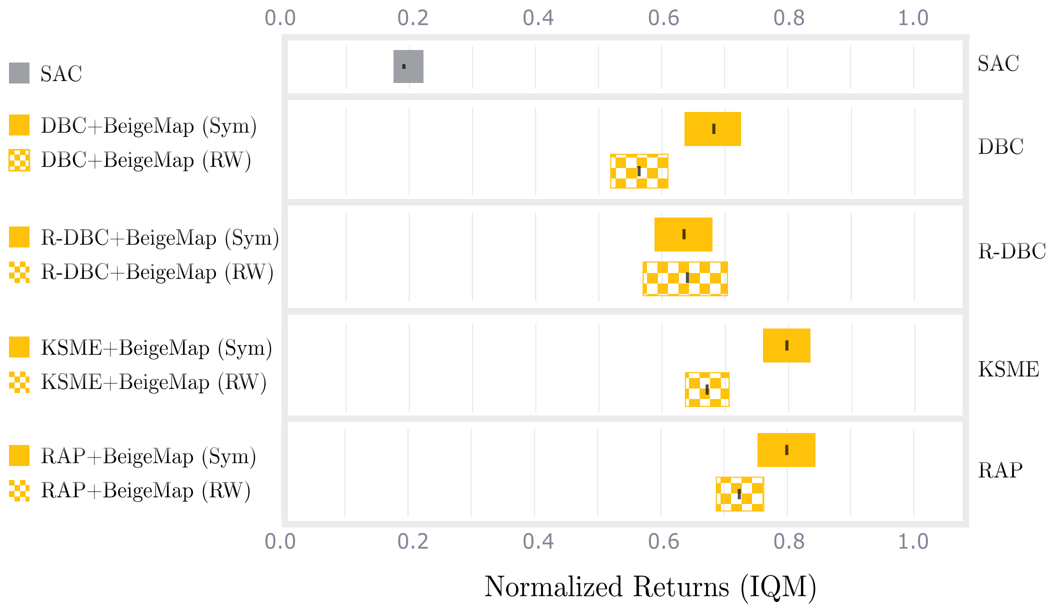

Figure 6: Interquartile median of total normalized returns of BeigeMap algorithms using the symmetric (Sym) normalization $K = \mathbb{D}^{-1/2}K\mathbb{D}^{-1/2}$ and $K = \mathbb{D}^{-1}K$. The symmetric normalization outperformed the random walk variant for all behavioral distances, which motivates our choice of the symmetric normalization for our experiments.

A.8. Additional Analysis

Probability of Improvement In Figure 7, we plot the probability of BeigeMap algorithms improving upon their baseline counterparts. Here, the probabilities of improvements correspond to the fraction of trials where one model obtained higher returns than the other. This measure does not consider the magnitude of differences, but illustrates the robustness of improvement from one algorithm over another (Agarwal et al., 2021). Improvements are considered significant if their confidence intervals do not contain 0.5. As shown in Figure 7, all models consistently lie above the 0.5 threshold, with the only exception of RAP+BeigeMap, whose confidence interval dips below the 0.5 threshold by a small margin. Overall, the addition of BeigeMaps is likely to improve the performance of current distance based representation learning.

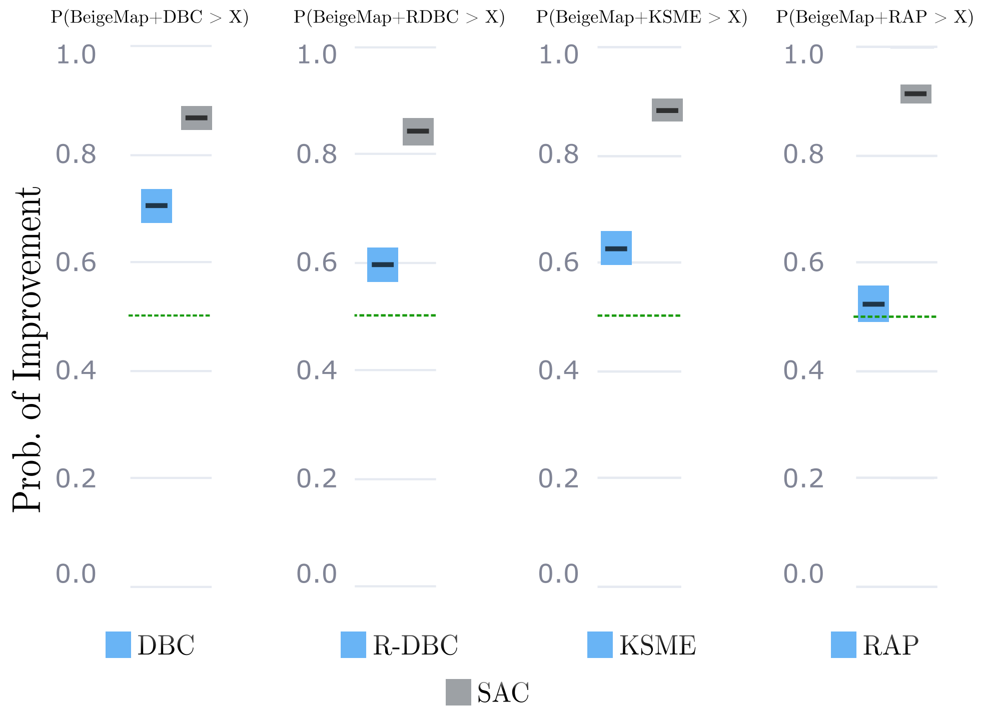

Figure 7: **Probabilities of Improvement** Likelihoods of BeigeMap algorithms outperforming their baselines.