DROGO: Default Representation Objective via Graph Optimization in Reinforcement Learning

Hon Tik Tse **Marlos C. Machado***
University of Alberta, Alberta Machine Intelligence Institute (Amii)
*Canada CIFAR AI Chair
{hontik, machado}@ualberta.ca

Abstract

The default representation (DR), originally introduced in neuroscience, and its principal eigenvector have been shown to be effective for a wide variety of applications, including reward shaping, count-based exploration, option discovery, and transfer. However, in prior investigations, the eigenvectors of the DR were computed by first approximating the DR matrix, and then performing an eigendecomposition. This procedure is computationally expensive and does not scale to high-dimensional spaces. In this paper, we propose an objective for approximating the principal eigenvector of the DR using only transitions with a neural network. This objective is inspired by a series of theoretical results and is empirically validated in a number of environments. We then demonstrate its usefulness by applying the learned eigenvectors for reward shaping.

1 Introduction

Learning representations is key to reinforcement learning (RL). The *successor representation* [SR; 1] is a promising temporal predictive representation, which represents a state as the expected discounted number of visits to successor states. In recent years, it has been shown to be effective in applications such as reward shaping [2, 3] and exploration [4, 5]. Beyond computational RL, evidence also suggests its relevance in human decision-making and neuroscience [6, 7, 8, 9].

The SR essentially isolates the temporal aspect of the problem from the reward. Given this property, once approximated, it can be used to perform flexible transfer to different reward functions. This idea underlies many recent transfer methods [10, 11]. However, for applications focusing on a single reward function, the SR can be limiting. Prior work [12] has shown that when applied to such applications, the SR's reward-agnostic nature can lead to undesirable behavior and poor performance.

One remedy is to use reward-aware representations for these applications. In particular, the default representation (DR), originally introduced in neuroscience to address the shortcomings of the SR [13], can be viewed as a reward-aware generalization of the SR (see Fig. 1). While the SR captures the number of transitions required to travel between states, the DR captures the rewards obtained when traveling between states. The DR has been shown to give rise to reward-aware behavior in various applications. For example, options induced by the DR take detours to avoid low-reward regions in the environment [12].

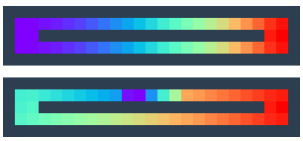

Figure 1: The principal eigenvectors of the SR (top) and DR (bottom) in a ring-shaped environment with a low-reward region at the top of the ring. The DR, unlike the SR, is reward-aware and captures the low-reward region in its principal eigenvector.

However, the study of the DR is limited to the tabular setting. While applications such as reward shaping and option discovery rely on the principal eigenvector of the DR, current methods

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

first approximate the full DR matrix before performing an eigendecomposition. Such a procedure is computationally expensive, and does not scale to environments with high-dimensional state spaces.

In this paper, we extend the DR to function approximation by proposing a novel objective, **D**efault **R**epresentation **O**bjective via **G**raph **O**ptimization (DROGO), which enables learning approximations of the principal eigenvector of the DR from samples. DROGO optimizes a variant of the graph-drawing objective [GDO; 14], where we reparameterize GDO in the log-space, and use an alternative constraint to control the norm of the learned vector. We derive the natural gradient [15, 16] to account for the inherent geometry of the log-space, which, left untreated, leads to significant instabilities during optimization. The alternative constraint, while empirically necessary for stability, comes at the expense of the theoretical guarantee that the objective retrieves the principal eigenvector. That is, DROGO approximates the principal eigenvector of the DR by approximating its terminal-state column, where the two vectors are not guaranteed to be equivalent. Nevertheless, we provide theoretical and empirical evidence demonstrating that the discrepancy between them is minimal.

In small, discrete domains where the ground-truth principal eigenvector can be retrieved for comparison, we empirically demonstrate that DROGO approximates the principal eigenvector of the DR extremely well. We apply DROGO to learn the principal eigenvector when using different state representations as input to the neural network, such as coordinates and pixels, demonstrating DROGO's robustness. Finally, we demonstrate the utility of the learned eigenvectors by applying them for reward shaping.

2 Preliminaries

In this paper, we use lowercase symbols (e.g., r, p) to denote functions or constants (e.g., c, ϵ), calligraphic font (e.g., $\mathcal{S}, \mathcal{A}$) to denote sets, bold lowercase symbols (e.g., $\mathbf{r}, \mathbf{e}$) to denote vectors, and bold uppercase symbols (e.g., $\mathbf{\Psi}, \mathbf{Z}$) to denote matrices. We index the (i, j)-th entry of a matrix $\mathbf{A}$ by $\mathbf{A}(i, j)$, and index the i-th entry of a vector $\mathbf{e}$ by $e(i)$.

2.1 Default Representation

We consider linearly solvable Markov decision processes [LS-MDPs; 17, 18] $(\mathcal{S}, \mathcal{A}, r, p, \pi_d)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \to \mathbb{R}_{<0}$ is the reward function, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, and π_d is the default policy. Let $\mathcal{S}_T \subset \mathcal{S}$ be the set of terminal states. We assume $\mathcal{S}_T$ is non-empty. We assume terminal states to be absorbing states and have a negative reward. We provide a more detailed discussion in Appendix A.

Let $\mathbf{r} \in \mathbb{R}_{<0}^{|\mathcal{S}|}$ be the vector of state rewards, and $\mathbf{P}^{\pi_d} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ be the transition probability matrix induced by π_d and p. The default representation [DR; 13, 12], defined with respect to π_d and denoted by $\mathbf{Z}$, is $\mathbf{Z} = [\mathrm{diag}(\exp(-\mathbf{r}/\lambda)) - \mathbf{P}^{\pi_d}]^{-1}$, where $\lambda > 0$ is a hyperparameter. We use $\mathbf{R}$ to denote $\mathrm{diag}(\exp(-\mathbf{r}/\lambda))$, and omit π_d as it is clear from context, so we write $\mathbf{Z} = (\mathbf{R} - \mathbf{P})^{-1}$. In practice, we symmetrize the DR by replacing $\mathbf{P}$ with $\tilde{\mathbf{P}} = (\mathbf{P} + \mathbf{P}^\top)/2$ to ensure a real eigensystem.

The DR has been applied in a variety of settings. In particular, its principal eigenvector (corresponding to the largest eigenvalue), $\mathbf{e}$, has been used for reward shaping and option discovery. However, the entries of $\mathbf{e}$ can have very small magnitudes, limiting its practical use without additional transformations. For example, simply applying $\mathbf{e}$ as a potential function for potential-based reward shaping [19] gives a shaping reward that is negligible almost everywhere. To address this, prior work [12] proved, for the discrete setting, that the principal eigenvector of the symmetrized DR is positive, and, in their experiments, used $\log \mathbf{e}$ instead, where $\log$ here denotes entry-wise logarithm.

2.2 Graph Drawing Objective

Spectral graph drawing [14] is a class of methods for visualizing graphs using the eigenvectors of matrices associated with the graph. These methods have been applied in RL since an MDP can be viewed as a graph, where the vertices are the states, and the edge weights are the transition probabilities between states induced by p and some policy π. Notably, the graph drawing objective (GDO) and its variants have been applied to retrieve the eigenvectors of the Laplacian, $\mathbf{L} = \mathbf{I} - \tilde{\mathbf{P}}^\pi$ [2, 3, 20]. Specifically, we can retrieve the smallest eigenvector (corresponding to the

smallest eigenvalue)[1] of a matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ through the following objective:

$$\min_{\mathbf{u}\in\mathbb{R}^{|\mathcal{S}|}} \mathbf{u}^\top \mathbf{M}\mathbf{u} \quad \text{s.t.} \quad \mathbf{u}^\top\mathbf{u} = c, \tag{1}$$

where $\mathbf{u}$ is the learned eigenvector, and c is a constant.

2.3 Natural Gradient

Natural gradient descent [15, 16] improves convergence properties over standard gradient descent by adjusting the update direction to account for the underlying structure of the parameter space.

Let $\mathcal{P} = \{\mathbf{w} \in \mathbb{R}^n\}$ be a parameter space, and the optimization problem be $\min_{\mathbf{w}} \ell(\mathbf{w})$, where ℓ is some loss function. Standard gradient descent makes the simplifying assumption that the distance between two points in $\mathcal{P}$ is Euclidean. The descent direction, $\delta\mathbf{w}$, is determined by the following optimization problem, where ϵ denotes some small constant:

$$\min_{\delta\mathbf{w}} \ell(\mathbf{w}) + \nabla\ell(\mathbf{w})^\top\delta\mathbf{w} \quad \text{s.t.} \quad \|\delta\mathbf{w}\|_2^2 = \epsilon^2. \tag{2}$$

That is, we search for the descent direction $\delta\mathbf{w}$ that locally minimizes ℓ, under the constraint that the Euclidean distance between the updated and original parameters is small. The descent direction $\delta\mathbf{w}$ is then proportional to $-\nabla\ell(\mathbf{w})$.

Natural gradient descent considers a more general notion of distance, where the distance between $\mathbf{w}$ and $\mathbf{w} + \delta\mathbf{w}$ is given by $\sqrt{\delta\mathbf{w}^\top\mathbf{G}(\mathbf{w})\delta\mathbf{w}}$. $\mathbf{G}(\mathbf{w})$ is known as the Riemannian metric tensor, which captures the local curvature of $\mathcal{P}$. Note that the Euclidean distance is a special case when $\mathbf{G} = \mathbf{I}$. The descent direction is now determined by

$$\min_{\delta\mathbf{w}} \ell(\mathbf{w}) + \nabla\ell(\mathbf{w})^\top\delta\mathbf{w} \;\text{ s.t. }\; \delta\mathbf{w}^\top\mathbf{G}(\mathbf{w})\delta\mathbf{w} = \epsilon^2. \tag{3}$$

While we still constrain the distance between the updated and original parameters, the distance now reflects the local structure of the parameter space. The descent direction is then $\propto -\mathbf{G}(\mathbf{w})^{-1}\nabla\ell(\mathbf{w})$.

3 DROGO

Prior work [12] adopts a computationally expensive procedure to obtain the logarithm of the principal eigenvector of the DR, $\log\mathbf{e}$. Recall that we are interested in the *logarithm* because the entries of $\mathbf{e}$ can be extremely small. Such an approach does not scale to high-dimensional spaces. To address this, we introduce an objective for approximating the logarithm of $\mathbf{e}$ using only transitions. We slowly motivate our objective through theory developed in several sections: Section 3.1 describes adapting the GDO for the DR. Section 3.2 describes log-space parameterization to approximate $\log\mathbf{e}$. Section 3.3 describes applying natural gradient to account for the log-space parameterization. Section 3.4 introduces an alternative constraint, which comes at the expense of theoretical guarantees. We conclude by introducing the final objective in Section 3.5.

3.1 Graph Drawing Objective

To learn the logarithm of the principal eigenvector, $\log\mathbf{e}$, of the symmetrized DR, $(\mathbf{R} - \tilde{\mathbf{P}})^{-1}$, a naive approach is to first learn $\mathbf{e}$ using the GDO, and then apply the logarithm. This is a necessary but not sufficient step towards our final objective, as we discuss below. To construct the GDO for the DR, we first have:

Proposition 3.1. *The principal eigenvector of* $(\mathbf{R} - \tilde{\mathbf{P}})^{-1}$ *is the smallest eigenvector of* $\mathbf{I} + \mathbf{R} - \tilde{\mathbf{P}}$.

Proof. Since a matrix and its inverse share the same eigenvectors in reverse order, the principal eigenvector of $(\mathbf{R} - \tilde{\mathbf{P}})^{-1}$ is equal to the smallest eigenvector of $\mathbf{R} - \tilde{\mathbf{P}}$. Furthermore, adding $\mathbf{I}$ to $\mathbf{R} - \tilde{\mathbf{P}}$ only shifts all eigenvalues by 1, and does not affect the set of eigenvectors or their order. □

[1] Note that this is equivalent to retrieving the principal eigenvector of the matrix's inverse, as is the case for the DR.

By Proposition 3.1, for some constant c, we can retrieve $\mathbf{e}$ by learning the smallest eigenvector of $\mathbf{I} + \mathbf{R} - \tilde{\mathbf{P}}$ using GDO[2]:

$$\min_{\mathbf{u} \in \mathbb{R}^{|\mathcal{S}|}} \mathbf{u}^\top (\mathbf{I} + \mathbf{R} - \tilde{\mathbf{P}}) \mathbf{u} \quad \text{s.t.} \quad \mathbf{u}^\top \mathbf{u} = c. \tag{4}$$

By inspecting Eq. 4, we gain intuition on what the principal eigenvector of the DR is in fact capturing. Note that $\mathbf{u}^\top (\mathbf{I} + \mathbf{R} - \tilde{\mathbf{P}}) \mathbf{u} = \mathbf{u}^\top (\mathbf{I} - \tilde{\mathbf{P}}) \mathbf{u} + \mathbf{u}^\top \mathbf{R} \mathbf{u}$, where the first term is just the objective for learning an eigenvector of the Laplacian (see Eq. 1), and the second term $\mathbf{u}^\top \mathbf{R} \mathbf{u} = \sum_s \exp(-r(s)/\lambda) u(s)^2$ can be viewed as a regularization term. Recall that $r(s)$ is negative. If a state s has a very negative reward, $\exp(-r(s)/\lambda)$ is large, and $u(s)$ is driven to have a small magnitude. The principal eigenvector of the DR can then be viewed as performing reward-aware regularization on top of an eigenvector of the Laplacian, where low-reward states are driven to have small magnitudes. Our empirical observations match this view.

As in prior work [2, 3, 20], the constraint $\mathbf{u}^\top \mathbf{u} = c$ can be turned into a quadratic penalty term, giving

$$\min_{\mathbf{u} \in \mathbb{R}^{|\mathcal{S}|}} \mathbf{u}^\top (\mathbf{I} + \mathbf{R} - \tilde{\mathbf{P}}) \mathbf{u} + b(\mathbf{u}^\top \mathbf{u} - c)^2, \tag{5}$$

where b is a hyperparameter. We denote the above objective function by $L_{\text{GDO}}(\mathbf{u})$.

Let $u_{\boldsymbol{\theta}}$ be the function approximated by a neural network parameterized by $\boldsymbol{\theta}$, and $\mathbf{u}_{\boldsymbol{\theta}} = [u_{\boldsymbol{\theta}}(s_1), \ldots, u_{\boldsymbol{\theta}}(s_{|\mathcal{S}|})]^\top$ be the vector predicted by the neural network. We have:

Proposition 3.2. *Given a transition (s, a, r, s') and an extra state s'', where s and s'' are sampled independently and uniformly at random, and $a \sim \pi_d$, we can minimize $L_{GDO}(\mathbf{u}_{\boldsymbol{\theta}})$ by minimizing*

$$\ell_{GDO}(\boldsymbol{\theta}) = \frac{1}{2}\big(u_{\boldsymbol{\theta}}(s) - u_{\boldsymbol{\theta}}(s')\big)^2 + \exp(-r/\lambda) u_{\boldsymbol{\theta}}(s)^2 + b'(u_{\boldsymbol{\theta}}(s)^2 - c')(u_{\boldsymbol{\theta}}(s'')^2 - c'), \tag{6}$$

where b' and c' can be treated as hyperparameters.

Proof sketch. We express $L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\theta}})$ in terms of expectations. ℓ_{GDO} is simply a sample-based approximation of the expectations. See Appendix B.1 for full proof.

We can compute ℓ_{GDO} using sampled transitions, then perform backpropagation and gradient descent to update $\boldsymbol{\theta}$. However, recall that some entries of $\mathbf{e}$ can have very small magnitudes. By optimizing Eq. 6, we observe that even small approximation errors can cause the neural network's prediction, $u_{\boldsymbol{\theta}}(s)$, to be negative for some low-reward states. This prohibits us from taking the logarithm and using the learned eigenvector in practice. While increasing λ can reduce the strength of the reward-aware regularization, extremely large λs are required to fully address this problem. However, this comes at the expense of the reward-awareness of $\mathbf{e}$, as $\mathbf{R} - \tilde{\mathbf{P}}$ approaches the Laplacian, $\mathbf{I} - \tilde{\mathbf{P}}$, as $\lambda \to \infty$.

3.2 Log-Space Parameterization

To guarantee that the learned eigenvector is always positive, while ensuring that it retains certain levels of reward-awareness, we perform log-space parameterization. Specifically, we directly approximate $\log \mathbf{e}$ using a neural network. First, Eq. 5 can be adapted as

$$\min_{\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}} \exp(\mathbf{v})^\top (\mathbf{I} + \mathbf{R} - \tilde{\mathbf{P}}) \exp(\mathbf{v}) + b(\exp(\mathbf{v})^\top \exp(\mathbf{v}) - c)^2, \tag{7}$$

where $\exp$ here denotes element-wise exponentiation. We denote the above objective function by $L^{\log}_{\text{GDO}}(\mathbf{v})$. Note that $L^{\log}_{\text{GDO}}(\mathbf{v}) = L_{\text{GDO}}(\exp(\mathbf{v}))$.

Let $v_{\boldsymbol{\phi}}$ be the function approximated by a neural network parameterized by $\boldsymbol{\phi}$. We define $\mathbf{v}_{\boldsymbol{\phi}}$ similar to how we defined $\mathbf{u}_{\boldsymbol{\theta}}$, and define $\mathbf{u}_{\boldsymbol{\phi}} = \exp(\mathbf{v}_{\boldsymbol{\phi}})$ to be the element-wise exponentiation of $\mathbf{v}_{\boldsymbol{\phi}}$.

Proposition 3.3. *Given a transition (s, a, r, s') and an extra state s'', where s and s'' are sampled independently and uniformly at random, and $a \sim \pi_d$, we can minimize $L^{\log}_{GDO}(\mathbf{v}_{\boldsymbol{\phi}})$ by minimizing*

$$\begin{aligned}\ell^{\log}_{GDO}(\boldsymbol{\phi}) = \frac{1}{2}\big(\exp(v_{\boldsymbol{\phi}}(s)) - \exp(v_{\boldsymbol{\phi}}(s'))\big)^2 + \exp(2v_{\boldsymbol{\phi}}(s) - r/\lambda) \\ + b(\exp(2v_{\boldsymbol{\phi}}(s)) - c)(\exp(2v_{\boldsymbol{\phi}}(s'')) - c).\end{aligned} \tag{8}$$

[2]For simplicity, we assume a uniform stationary distribution throughout the paper. Note that the objectives trivially generalize to π_d-induced stationary distributions, e.g., the inner product $\mathbf{u}^\top \mathbf{u}$ will be defined w.r.t. such distributions.

Proof. We replace $u_{\boldsymbol{\theta}}(s)$ with $\exp(v_{\boldsymbol{\phi}}(s))$ in Eq. 6. □

While it is possible to just approximate $\ell^{\log}_{\text{GDO}}$ and perform backpropagation and gradient descent to update $\boldsymbol{\phi}$, we perform further analysis on the gradient of $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$. By comparing the gradient of $L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\theta}})$ and $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$, we reveal a source of instability for the loss function, $\ell^{\log}_{\text{GDO}}$.

Using the multivariate chain rule, the gradient of $L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\theta}})$ w.r.t. the i-th entry of the parameters, $\boldsymbol{\theta}$, is

$$\frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\theta}})}{\partial \theta_i} = \frac{\partial \mathbf{u}_{\boldsymbol{\theta}}}{\partial \theta_i} \cdot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\theta}})}{\partial \mathbf{u}_{\boldsymbol{\theta}}}, \tag{9}$$

while the gradient of $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$ w.r.t. the i-th entry of $\boldsymbol{\phi}$ is

$$\frac{\partial L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})}{\partial \phi_i} = \frac{\partial \mathbf{v}_{\boldsymbol{\phi}}}{\partial \phi_i} \cdot \frac{\partial L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})}{\partial \mathbf{v}_{\boldsymbol{\phi}}} = \frac{\partial \mathbf{v}_{\boldsymbol{\phi}}}{\partial \phi_i} \cdot \left(\exp(\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial \overbrace{L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})}^{L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}}{\partial \mathbf{u}_{\boldsymbol{\phi}}} \right) \tag{10}$$

where $\odot$ is the Hadamard product. Comparing Eq. 9 and 10, we see that the log-space parameterization introduces an extra multiplicative exponential term $\exp(\mathbf{v}_{\boldsymbol{\phi}})$, which scales exponentially with the network output, and can lead to instability in the optimization process.

3.3 Natural Gradient

When performing log-space parameterization, we essentially move from solving $\min_{\mathbf{u}} L(\mathbf{u})$ to solving $\min_{\mathbf{v}} L(\exp(\mathbf{v}))$, and now we have $\mathbf{v}$ as parameters of the optimization problem instead of $\mathbf{u}$. When performing standard gradient descent, we are implicitly assuming that the difference between the original and updated parameters is measured by their Euclidean distance, or equivalently, the size of the gradient step is measured by its Euclidean norm. While this assumption generally works without issues, it can become undesirable when we perform log-space parameterization. Suppose we take a gradient step with a tiny Euclidean norm. When $\mathbf{v}$ has large values, the gradient step causes massive changes in $\exp(\mathbf{v})$, and thus, $L(\exp(\mathbf{v}))$. When $\mathbf{v}$ has small values, the same gradient step causes negligible changes. This is the reason we observe the exponential term, $\exp(\mathbf{v}_{\boldsymbol{\phi}})$, in Eq. 10. To summarize, the $\mathbf{v}$-space's local geometry is inherently different from that of the $\mathbf{u}$-space.

We can address this problem with natural gradient, by explicitly accounting for the difference in the geometry of the $\mathbf{v}$-space. To do so, we need to identify the distance in the $\mathbf{v}$-space induced by: (1) the relationship between the $\mathbf{v}$- and $\mathbf{u}$-spaces, and (2) the Euclidean assumption on the $\mathbf{u}$-space (see Figure 4, in the Appendix, for a visual illustration).

Proposition 3.4. *The distance in the* $\mathbf{v}$*-space is characterized by the Riemannian metric tensor* $\mathbf{G}(\mathbf{v}) = \operatorname{diag}(\exp(2\mathbf{v}))$.

Proof sketch. Let $\delta\mathbf{u}$ be the change induced in $\mathbf{u}$ by $\delta\mathbf{v}$, a small step in the $\mathbf{v}$-space. We identify $\mathbf{G}(\mathbf{v})$ by solving $d_{\mathcal{U}}(\mathbf{u}, \mathbf{u}+\delta\mathbf{u})^2 = d_{\mathcal{V}}(\mathbf{v}, \mathbf{v}+\delta\mathbf{v})^2$, where $d_{\mathcal{U}}$ and $d_{\mathcal{V}}$ are distances in the $\mathbf{u}$- and $\mathbf{v}$-spaces respectively. We provide the full proof in Appendix B.2.

Corollary 3.5. *Let* $-\frac{\partial L}{\partial \mathbf{v}}$ *be the standard gradient descent direction. The adjusted natural gradient descent direction is simply* $-\mathbf{G}(\mathbf{v})^{-1}\frac{\partial L}{\partial \mathbf{v}} = -\operatorname{diag}(\exp(-2\mathbf{v}))\frac{\partial L}{\partial \mathbf{v}}$.

We apply the natural gradient to minimize $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$. Incorporating the Riemannian metric tensor to Eq. 10, the gradient of $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$ w.r.t. the i-th entry of the parameters, $\boldsymbol{\phi}$, is

$$\frac{\partial \mathbf{v}_{\boldsymbol{\phi}}}{\partial \phi_i} \cdot \mathbf{G}(\mathbf{v}_{\boldsymbol{\phi}})^{-1} \left(\exp(\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}} \right) = \frac{\partial \mathbf{v}_{\boldsymbol{\phi}}}{\partial \phi_i} \cdot \left(\exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}} \right). \tag{11}$$

Note that Eq. 10 and 11 differ only by the sign inside the exponential. Now, $\exp(-\mathbf{v}_{\boldsymbol{\phi}})$ ensures that updates to $\mathbf{v}_{\boldsymbol{\phi}}$ produce similar effects on $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$ regardless of the value $\mathbf{v}_{\boldsymbol{\phi}}$ takes. For example, if $v_{\boldsymbol{\phi}}(s)$ is very large, for which even a tiny update can drastically change $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$, $\exp(-\mathbf{v}_{\boldsymbol{\phi}})$ scales the update to $v_{\boldsymbol{\phi}}(s)$ by $1/\exp(v_{\boldsymbol{\phi}}(s))$.

Using the above equation, we now derive a new loss function that incorporates the natural gradient. Note that Eq. 11 is the dot product of two terms, where the first term is the gradient of the neural network's output w.r.t. its parameters, and the second term is the natural gradient of the objective function, $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$, with respect to the neural network's output. We present the following:

Proposition 3.6. *Given a transition (s, a, r, s') and an extra state s'', where s and s'' are sampled independently and uniformly at random, and $a \sim \pi_d$, we can minimize $L^{\log}_{GDO}(\mathbf{v}_{\boldsymbol{\phi}})$, with hyperparameters λ, b'', c'', with natural gradient descent by minimizing*

$$\ell^{NG}_{GDO}(\boldsymbol{\phi}) = [\![\exp(-r/\lambda) - \exp(v_{\boldsymbol{\phi}}(s') - v_{\boldsymbol{\phi}}(s)) + b''(\exp(2v_{\boldsymbol{\phi}}(s'')) - c'')]\!]v_{\boldsymbol{\phi}}(s), \tag{12}$$

where $[\![\cdot]\!]$ denotes the stop-gradient operator.

Proof sketch. We derive a sampled-based approximation of Eq. 11. It is trivial to show that the gradient of $\ell^{\text{NG}}_{\text{GDO}}$ recovers this approximation. We provide the full proof in Appendix B.3.

When using $\ell^{\text{NG}}_{\text{GDO}}$, we are able to successfully learn $\log \mathbf{e}$ in the tabular case (see Appendix D.4), and when using one-hot representations of states as input to the neural network. However, we observe instabilities during optimization when using other state representations, which stems from the difficulty in balancing the GDO, $\exp(\mathbf{v})^\top(\mathbf{I}+\mathbf{R}-\tilde{\mathbf{P}})\exp(\mathbf{v})$, and the quadratic norm penalty, $(\exp(\mathbf{v})^\top \exp(\mathbf{v}) - c)^2$. In $\ell^{\text{NG}}_{\text{GDO}}(\boldsymbol{\phi})$, b'' is the hyperparameter balancing the two terms. We observe empirically that when b'' is too small, the norm penalty is not strong enough, and the norm of the learned $\exp(\mathbf{v}_{\boldsymbol{\phi}})$ can vary significantly over the course of learning. When b'' is too large, the norm penalty constrains the norm of $\exp(\mathbf{v}_{\boldsymbol{\phi}})$ well, but interferes with the objective. We provide more details in Appendix D.5.

In the next section, we introduce the final adaptation to obtain our final loss function. It differs from $\ell^{\text{NG}}_{\text{GDO}}$ by how we constrain the learned vector, given our observation that the quadratic norm penalty can be difficult to balance.

3.4 Alternative Constraint on the Learned Vector

Despite being difficult to balance, the quadratic norm penalty prevents the learned eigenvector from collapsing to the zero vector. An alternative way of seeing this is to imagine that, without the quadratic norm penalty, a learned vector perfectly aligned with the ground-truth can be expressed as $\alpha\mathbf{e}$ for some α, and in this case the objective function is minimized when $\alpha \to 0$. We still have the same issue when learning $\log \mathbf{e}$ directly instead of $\mathbf{e}$, as multiplying $\mathbf{e}$ by a positive constant α would be equivalent to adding the constant $\log \alpha$ to all entries of $\log \mathbf{e}$. When $\alpha \to 0$, all entries of the solution keep drifting to the left on the real number line.

Thus, to drop the quadratic norm penalty, we need to prevent such constant shifts. We do so by fixing the value of a particular entry of the learned vector. In particular, we fix the value of a selected terminal state, $v(s_T)$, to 0. Intuitively, $v(s_T)$ acts as an anchor that holds the entire vector $\mathbf{v}$ in place. Mathematically, for a terminal state $s_T \in \mathcal{S}_T$, we solve the following optimization problem:

$$\min_{\mathbf{v}\in\mathbb{R}^{|\mathcal{S}|}} \exp(\mathbf{v})^\top(\mathbf{R} - \tilde{\mathbf{P}})\exp(\mathbf{v}) \text{ s.t. } v(s_T) = 0. \tag{13}$$

While Eq. 13 might make intuitive sense, we in fact lose the theoretical guarantee that we are directly learning the smallest eigenvector of $\mathbf{R} - \tilde{\mathbf{P}}$. We first study the solution of Eq. 13, and present:

Proposition 3.7. *Minimizing Eq. 13 retrieves the logarithm of the column of the DR corresponding to the selected terminal state, s_T.*

Proof. Using Lagrange multipliers, Eq. 13 is equivalent to

$$\min_{\mathbf{v}\in\mathbb{R}^{|\mathcal{S}|}} \max_{\mu\in\mathbb{R}} \exp(\mathbf{v})^\top(\mathbf{R} - \tilde{\mathbf{P}})\exp(\mathbf{v}) + \mu v(s_T). \tag{14}$$

Let $\mathbf{b}_i$ be the standard basis vector with a 1 at entry i. Setting the gradient w.r.t. $\mathbf{v}$ to 0, we have

$$2\exp(\mathbf{v}) \odot (\mathbf{R} - \tilde{\mathbf{P}})\exp(\mathbf{v}) + \mu\mathbf{b}_{s_T} = 0 \tag{15}$$

$$(\mathbf{R} - \tilde{\mathbf{P}})\exp(\mathbf{v}) = -\frac{\mu}{2}\exp(-v(s_T))\mathbf{b}_{s_T} \tag{16}$$

$$\exp(\mathbf{v}) = -\frac{\mu}{2}(\mathbf{R} - \tilde{\mathbf{P}})^{-1}\mathbf{b}_{s_T}, \tag{17}$$

where $(\mathbf{R}-\tilde{\mathbf{P}})^{-1}\mathbf{b}_{s_T}$ is just the column of the DR corresponding to the chosen terminal state, s_T. □

Note that we also extend Proposition 3.7 to the case of multiple terminal states in Appendix B.4. Theoretical analysis reveals that by solving Eq. 13, we in fact retrieve the s_T column of the DR. Nevertheless, we observe empirically that the vector we learn aligns with the principal eigenvector of the DR very well. We provide a partial explanation by proving the following theorem:

Theorem 3.8. *Let $r(s_T) = -\delta \; \forall s_T \in \mathcal{S}_T$, where $\delta > 0$. When $\delta \to 0^+$, the columns of the DR ($(\mathbf{R} - \mathbf{P})^{-1}$) corresponding to all s_T form a basis for the eigenspace of its principal eigenvalue.*

Proof sketch. We first show that δ is the principal eigenvalue of the DR. We derive the DR columns corresponding to terminal states, and the eigenvectors corresponding to δ. We conclude the theorem by comparing their formulae. We provide the full proof in Appendix B.5.

Theorem 3.8 states that any terminal-state column of the DR is a principal eigenvector of the DR. However, Theorem 3.8 considers the DR without symmetrization, while in Eq. 13, we use the symmetrized DR, where we replace $\mathbf{P}$ with $\tilde{\mathbf{P}} = (\mathbf{P} + \mathbf{P}^\top)/2$. Nevertheless, it provides good intuition for why optimizing Eq. 13 retrieves a vector that highly aligns with the principal eigenvector of the symmetrized DR.

Remark 3.9. Theorem 3.8, combined with the fact that the s-th column of the DR captures the reward-aware distance from all states to s [12], explains why the principal eigenvector of the DR has been successful in applications such as reward shaping.

As Theorem 3.8 merely provides a partial, intuitive argument, we now directly bound the angle between the terminal-state column and the principal eigenvector of the symmetrized DR.

Theorem 3.10. *When there is a single terminal state, the angle between the terminal-state column and the principal eigenvector of the symmetrized DR, θ, satisfies*

$$\tan\theta \leq \frac{|\mu_1|}{|\mu_2||w_1(s_T)|}, \tag{18}$$

where μ_1, μ_2 are the smallest and second smallest eigenvalues of $\mathbf{R} - \tilde{\mathbf{P}}$ respectively, and $w_1(s_T)$ is the entry of the principal eigenvector of the symmetrized DR corresponding to the terminal state.

Proof sketch. We express the terminal-state column as the sum of two components: (1) one along the direction of the principal eigenvector, and (2) one orthogonal to the principal eigenvector. We then use the norm of the two components to compute an upper bound on θ. See Appendix B.6 for the full proof, where we also provide the bound for the case of multiple terminal states.

As we demonstrate in Appendix C, for the symmetrized DR, the discrepancy between the terminal-state column and the principal eigenvector is minimal for the set of environments we consider in our experiments, and the terminal-state column remains a close approximation of the principal eigenvector.

3.5 DROGO

Inspired by the theoretical results presented so far, we present the **D**efault **R**epresentation **O**bjective via **G**raph **O**ptimization (DROGO), an objective that approximates $\log \mathbf{e}$ by optimizing Eq. 13 using natural gradient. Let $v_{\boldsymbol{\phi}}$ be the function approximated by a neural network parameterized by $\boldsymbol{\phi}$.

Theorem 3.11 (DROGO loss function)**.** *Let $\mathbb{I}$ denote the indicator function. Given a transition (s, a, r, s'), where s is sampled uniformly at random, and $a \sim \pi_d$, we minimize Eq. 13 by minimizing*

$$\ell_{DROGO}(\boldsymbol{\phi}) = \Big[\!\Big[\exp(-r/\lambda) - \exp\Big(v_{\boldsymbol{\phi}}(s')(1 - \mathbb{I}_{\{s'=s_T\}}) - v_{\boldsymbol{\phi}}(s)\Big)\Big]\!\Big] v_{\boldsymbol{\phi}}(s). \tag{19}$$

Proof. Removing the quadratic penalty, Eq. 12 becomes

$$[\![\exp(-r/\lambda) - \exp(v_{\boldsymbol{\phi}}(s') - v_{\boldsymbol{\phi}}(s))]\!] v_{\boldsymbol{\phi}}(s). \tag{20}$$

The constraint $v(s_T) = 0$ can be enforced simply by multiplying $v_{\boldsymbol{\phi}}(s')$ by $(1 - \mathbb{I}_{\{s'=s_T\}})$. □

4 Experiments

We demonstrate empirically that optimizing the DROGO objective approximates the principal eigenvector of the DR extremely well. We present a comparison of DROGO and the intermediate objectives in Appendix D.4. We also study the sensitivity of DROGO to λ (in Eq. 19) in Appendix D.3. Finally, we show the utility of the learned eigenvectors for reward shaping.

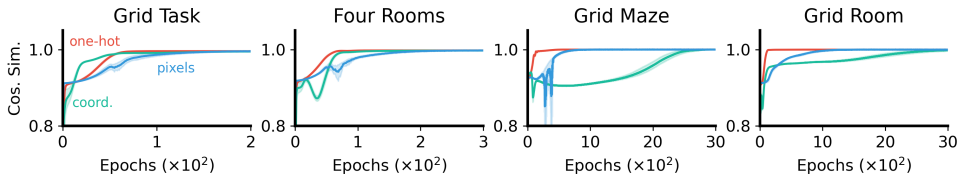

Figure 2: Cosine similarity between the learned and ground-truth principal eigenvectors for different state representations averaged over 10 seeds. Shaded area indicates 95% bootstrap confidence interval.

4.1 Learning the Principal Eigenvector of the DR

We apply DROGO to approximate the principal eigenvector of the DR using neural networks. A neural network takes as input a state, and approximates the logarithm of the corresponding entry in the principal eigenvector. To be able to validate our results against the ground-truth, we consider the set of grid world environments in [12] (see Appendix D.1). We also demonstrate the applicability of DROGO in continuous state spaces by applying it in point maze (see Appendix E).

We generate 200,000 transitions using a uniform random initial state distribution and the default policy, chosen to be the uniform random policy. We then train the neural networks by minimizing the DROGO loss (Eq. 19) using mini-batches from the dataset. To demonstrate the robustness of DROGO, we use three different representations of states as input to the neural network: (1) one-hot encoding, where state s_i is represented as $\mathbf{b}_i$, (2) (x, y)-coordinates, and (3) pixels. See details in Appendix D.2.

To evaluate the quality of the approximation, we compute the cosine similarity between the learned vector and the ground-truth. The cosine similarity is computed between the logarithm of these eigenvectors. Fig. 2 shows the cosine similarity in four grid world environments for the three state representations. We can see that regardless of the choice of the state representation, DROGO enables approximating the principal eigenvector very well, as the cosine similarity converges to near 1 for all settings. A visual comparison of the learned and ground-truth eigenvectors is in Appendix D.2.

4.2 Reward Shaping

We now demonstrate that the information contained in these learned vectors can be extracted and transformed into performance gains, and that pathological issues do not arise from inevitable approximation errors. To do so, similar as prior work [2, 3], we apply the learned vectors for reward shaping.

In reward shaping, we provide the agent with a shaping reward to aid in learning the optimal policy. We apply the principal eigenvector of the DR for potential-based reward shaping [19]. Let $\mathbf{v}$ be the learned vector, the shaping reward is $\tilde{r}_t = \gamma v(s_{t+1}) - v(s_t)$, where γ is the discount factor. Then, we train a Q-learning [21] agent using a convex combination of the original environment reward and the shaping reward, resulting in $(1-\beta)r_t + \beta\tilde{r}_t$, where $\beta \in [0, 1]$ is a hyperparameter.

We perform reward shaping using the approximated principal eigenvector of the DR in separate experiments for each state representation. We compare against using the principal eigenvectors of the SR [1], the first-occupancy representation [FR; 22], and the λ-representation with $\lambda = 0.5$ [λR; 23], using the shortest distance from the goal as the shaping reward (DISTANCE), and no reward shaping (plain Q-learning). Applying the principal eigenvectors of the SR and DR for reward shaping is common in RL [2, 3, 12]. We compute the principal eigenvector of the SR, FR, and λR in closed form to prevent approximation errors from hurting the performance of the baseline, effectively giving them an advantage. Similar to [12], we use $\gamma = 0.99$, $\epsilon = 0.05$ for ϵ-greedy exploration, and perform a grid search over β ($[0.25, 0.5, 0.75, 1.0]$), and Q-learning's step size ($[0.1, 0.3, 1.0]$).

We perform reward shaping in the same environments as Section 4.1. In these environments, the agent is tasked with reaching the goal state while avoiding low-reward regions on the way. To measure the effectiveness of different shaping rewards, we evaluate the mean number of steps required to converge to the optimal return (N_{OPT}). As the DR has been shown to give rise to reward-aware behavior, we also measure the mean number of visits to low-reward states during training (N_{VISIT}). The lower N_{OPT}, the more effective the shaping reward. The lower N_{VISIT}, the more the shaping reward guides the agent to avoid low-reward regions.

Table 1: The mean number of steps, over 10 seeds, required to converge to the optimal return (N_{OPT}). The lower- and superscripts denote the 95% bootstrap confidence interval. We highlight all methods whose confidence intervals overlap with the lowest mean.

ENV	NS	SR	λR$_{0.5}$	FR	DISTANCE	DR$_{\text{GT}}$	DR$_{\text{ONE-HOT}}$	DR$_{\text{COORD}}$	DR$_{\text{PIXELS}}$
GRID TASK	$13.75^{+0.17}_{-0.18}$	$2.04^{+0.04}_{-0.04}$	$1.38^{+0.13}_{-0.09}$	$2.12^{+0.34}_{-0.23}$	-	$\mathbf{1.54^{+0.78}_{-0.59}}$	$\mathbf{0.97^{+0.04}_{-0.04}}$	$\mathbf{0.98^{+0.04}_{-0.05}}$	$\mathbf{0.98^{+0.05}_{-0.05}}$
FOUR ROOMS	$14.39^{+0.27}_{-0.19}$	$1.37^{+0.04}_{-0.04}$	$5.32^{+1.05}_{-1.12}$	$2.93^{+0.79}_{-0.55}$	$4.97^{+1.09}_{-1.00}$	$\mathbf{0.16^{+0.01}_{-0.00}}$	$\mathbf{0.18^{+0.04}_{-0.03}}$	$\mathbf{0.16^{+0.00}_{-0.00}}$	$1.25^{+1.83}_{-1.08}$
GRID MAZE	$71.82^{+4.29}_{-3.32}$	$15.20^{+0.37}_{-0.34}$	$28.43^{+0.53}_{-0.52}$	$30.03^{+0.66}_{-0.54}$	$19.71^{+9.51}_{-6.73}$	$\mathbf{5.50^{+0.19}_{-0.18}}$	$\mathbf{5.97^{+0.29}_{-0.28}}$	$\mathbf{6.46^{+1.14}_{-0.72}}$	$7.47^{+1.63}_{-1.45}$
GRID ROOM	-	-	$33.95^{+0.22}_{-0.24}$	$38.76^{+0.14}_{-0.12}$	-	$\mathbf{8.00^{+0.22}_{-0.22}}$	$\mathbf{7.68^{+0.17}_{-0.17}}$	$\mathbf{7.72^{+0.25}_{-0.32}}$	$\mathbf{7.83^{+0.22}_{-0.24}}$

Table 2: The mean number of visits to low-reward states (N_{VISIT}), over 10 seeds. The lower- and superscripts denote the 95% bootstrap confidence interval. We highlight all methods whose confidence intervals overlap with the lowest mean.

ENV	NS	SR	λR$_{0.5}$	FR	DISTANCE	DR$_{\text{GT}}$	DR$_{\text{ONE-HOT}}$	DR$_{\text{COORD}}$	DR$_{\text{PIXELS}}$
GRID TASK	$7.87^{+0.12}_{-0.12}$	$6.51^{+0.49}_{-0.43}$	$7.91^{+0.19}_{-0.22}$	$6.41^{+0.25}_{-0.31}$	$7.68^{+0.29}_{-0.21}$	$\mathbf{5.17^{+0.13}_{-0.13}}$	$6.62^{+0.59}_{-0.61}$	$6.61^{+0.87}_{-0.74}$	$6.60^{+0.70}_{-0.55}$
FOUR ROOMS	$3.12^{+0.07}_{-0.08}$	$1.30^{+0.03}_{-0.03}$	$1.24^{+0.04}_{-0.04}$	$1.19^{+0.08}_{-0.08}$	$1.30^{+0.07}_{-0.07}$	$\mathbf{0.06^{+0.02}_{-0.03}}$	$\mathbf{0.05^{+0.03}_{-0.03}}$	$\mathbf{0.07^{+0.02}_{-0.03}}$	$\mathbf{0.06^{+0.04}_{-0.03}}$
GRID MAZE	$0.65^{+0.03}_{-0.03}$	$\mathbf{0.42^{+0.04}_{-0.04}}$	$\mathbf{0.43^{+0.04}_{-0.04}}$	$\mathbf{0.41^{+0.03}_{-0.03}}$	$0.50^{+0.04}_{-0.05}$	$\mathbf{0.47^{+0.04}_{-0.04}}$	$\mathbf{0.44^{+0.02}_{-0.02}}$	$\mathbf{0.46^{+0.03}_{-0.03}}$	$\mathbf{0.46^{+0.03}_{-0.03}}$
GRID ROOM	$1.47^{+0.05}_{-0.07}$	$1.73^{+0.07}_{-0.07}$	$0.62^{+0.03}_{-0.03}$	$0.93^{+0.08}_{-0.09}$	$0.88^{+0.05}_{-0.06}$	$\mathbf{0.59^{+0.04}_{-0.06}}$	$\mathbf{0.53^{+0.03}_{-0.03}}$	$\mathbf{0.55^{+0.05}_{-0.04}}$	$\mathbf{0.54^{+0.03}_{-0.03}}$

Table 1 and 2 present N_{OPT} and N_{VISIT} for different shaping rewards respectively, averaged over 10 seeds. In the grid room environment, the SR and no reward shaping cannot converge to the optimal return by the end of training, so the corresponding N_{OPT}'s are not reported. The same is true for DISTANCE in grid task and grid room. Using the DR for reward shaping consistently leads to lower N_{OPT} compared to the baselines. We observe that when using the DR with pixels representation for reward shaping in four rooms, there are a small number of seeds with slower learning speed due to small approximation errors in the principal eigenvector, leading to larger N_{OPT} than its one-hot and coordinates counterparts. Nevertheless, it achieves comparable performance to the SR, whose principal eigenvector is computed in closed form and does not have to contend with approximation errors. On the other hand, using the DR for reward shaping leads to similar or lower N_{VISIT} compared to the SR, FR, and λR. The difference is particularly significant in four rooms and grid room.

In general, the DR performs better than the SR, FR, λR, and DISTANCE. This is because, unlike these baselines, the DR captures reward information in the principal eigenvector. During reward shaping, the reward-aware principal eigenvector causes the agent to prefer high-reward paths in the environment, eventually leading to lower N_{OPT} and N_{VISIT}. These results match the results reported by Tse et al. [12] in the tabular case, but now using eigenvectors approximated through neural networks obtained from high-dimensional observations. This is a substantial improvement over computing such eigenvectors in closed form in the tabular case. Finally, we also provide learning curves in Appendix D.6.

5 Related Work

The SR [1] represents a state by the expected discounted number of visits to successor states. Specifically, it counts all visits to successor states. The first-occupancy representation [FR; 22] is a variant that captures only the first visit to successor states. The λ representation [λR; 23] can be viewed as an interpolation between the SR and FR. Compared to the FR and λR, the DR [13] generalizes the SR in an orthogonal direction by capturing not just the transitions, but also the rewards.

The SR and its extension in continuous spaces, the successor measure [24], have given rise to a wide range of methods for transfer. Such works include successor features [10, 25, 26], forward-backward representation [11, 27, 28], proto-successor measure [29], and distributional successor measure [30]. The DR, unlike the SR, also captures the reward function. Being more specific to environment dynamics comes at the price of less flexible transfer. Without assuming access to environment dynamics, it can be used to directly compute the optimal policy when only rewards at terminal states change [12].

Apart from transfer, where the goal is to efficiently compute the optimal policy for downstream rewards, the SR has been applied to better learn from a single reward function. The SR has the same set of eigenvectors as the Laplacian [31, 7], which was shown to be a good basis for

approximating the value function [32, 33]. As they reflect the geometry of the state space, they have been subsequently applied as a distance metric for reward shaping [2, 3, 34], an inductive bias for temporally-extended exploration [5, 35], effective state representations [36, 37], and a density model for count-based exploration [4]. Given the close connection between the SR and DR, Tse et al. [12] applied the principal eigenvector of the DR to similar applications, and found that the DR gave rise to better performance than the SR due to its reward-aware nature.

The graph-drawing objective [GDO; 14] has been applied to learn the eigenvectors of the SR (or Laplacian) in RL [2]. Extensions address problems with the GDO such as arbitrary rotations of the learned eigenvectors [3, 20]. However, similar methods do not exist for the DR. DROGO is the first to approximate the principal eigenvector of the DR.

Apart from its application in computational RL, since the SR can be viewed as a middle ground between model-free and model-based decision making, it has received interest in behavioral and neuroscience studies [6, 7, 8, 9]. The DR was proposed originally in neuroscience to address, among others, the inflexibility of the SR in replanning tasks [13], and has been extended recently to incorporate compositionality [38]. Different from these works, we focus solely on computational RL.

6 Conclusion

In this paper, we present DROGO, which optimizes a variant of the GDO with log-space parameterization and a point-wise constraint using natural gradient. Empirically, we demonstrate the effectiveness of DROGO in a number of environments with different state representations. We further demonstrate the utility of the learned eigenvectors by performing reward shaping.

In this work, our experiments focus on demonstrating that DROGO effectively approximates the principal eigenvector of the DR. To demonstrate that inevitable approximation errors with neural networks do not lead to pathological behaviors, we applied the learned vectors in reward shaping with a two-stage approach, where we first learn the vector before applying it for reward shaping. Such experiments serve our stated purpose, and should not be misconstrued as demonstrations of how the approximated vectors should be applied in practice. Our experiments have also mostly focused on small, discrete domains, and the scalability to genuinely high-dimensional visual and continuous-control domains remains to be demonstrated. Another limitation of this work is that DROGO, limited by the LS-MDPs framework in which the DR is defined, can only be applied to environments with terminal states.

Future work can focus on demonstrating the utility of these learned vectors in more practical scenarios, such as approximating both the principal eigenvector and the policy simultaneously, or applying the learned vectors for more complex downstream applications, similar to how prior work [35, 39] used GDO [3, 20] for exploration and generalization in RL. Given the recent prominence of Laplacian-/SR-based research in RL, DROGO unlocks promising new directions by scaling up a reward-aware generalization of the SR.

Acknowledgments and Disclosure of Funding

We thank Diego Gomez and Siddarth Chandrasekar for useful discussions, and Martha White for feedback on the manuscript. We thank anonymous reviewers whose feedback improved this work. This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chair Program, and Alberta Innovates. It was also enabled in part by computational resources provided by the Digital Research Alliance of Canada.

References

[1] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[2] Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *International Conference on Learning Representations (ICLR)*, 2019.

[3] Kaixin Wang, Kuangqi Zhou, Qixin Zhang, Jie Shao, Bryan Hooi, and Jiashi Feng. Towards better Laplacian representation in reinforcement learning with generalized graph drawing. In *International Conference on Machine Learning (ICML)*, 2021.

[4] Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[5] Marlos C. Machado, Andre Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research*, 24(80):1–69, 2023.

[6] Ida Momennejad, Evan M. Russek, Jin H. Cheong, Matthew M. Botvinick, Nathaniel D. Daw, and Samuel J. Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680–692, 2017.

[7] Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. Design principles of the hippocampal cognitive map. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[8] Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, 2017.

[9] Diego Gomez, Michael Bowling, J Quinn Lee, and Marlos C Machado. Representations in the hippocampal-entorhinal system emerge from learning sensory predictions. *bioRxiv*, 2025.

[10] André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[11] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[12] Hon Tik Tse, Siddarth Chandrasekar, and Marlos C. Machado. Reward-aware proto-representations in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[13] Payam Piray and Nathaniel D. Daw. Linear reinforcement learning in planning, grid fields, and cognitive control. *Nature Communications*, 12(1):4942, 2021.

[14] Yehuda Koren. On spectral graph drawing. In *International Computing and Combinatorics Conference*, 2003.

[15] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276, 1998.

[16] Shun-Ichi Amari and Scott C Douglas. Why natural gradient? In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1998.

[17] Emanuel Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

[18] Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009.

[19] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, 1999.

[20] Diego Gomez, Michael Bowling, and Marlos C. Machado. Proper Laplacian representation learning. In *International Conference on Learning Representations (ICLR)*, 2024.

[21] Christopher Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[22] Ted Moskovitz, Spencer R Wilson, and Maneesh Sahani. A first-occupancy representation for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[23] Ted Moskovitz, Samo Hromadka, Ahmed Touati, Diana Borsa, and Maneesh Sahani. A state representation for diminishing rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[24] Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *CoRR*, abs/2101.07123, 2021.

[25] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning (ICML)*, 2018.

[26] Diana Borsa, André Barreto, John Quan, Daniel J. Mankowitz, Hado van Hasselt, Rémi Munos, David Silver, and Tom Schaul. Universal successor features approximators. In *International Conference on Learning Representations (ICLR)*, 2019.

[27] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *International Conference on Learning Representations (ICLR)*, 2023.

[28] Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *International Conference on Learning Representations (ICLR)*, 2025.

[29] Siddhant Agarwal, Harshit Sikchi, Peter Stone, and Amy Zhang. Proto successor measure: Representing the behavior space of an RL agent. In *International Conference on Machine Learning (ICML)*, 2025.

[30] Harley Wiltzer, Jesse Farebrother, Arthur Gretton, Yunhao Tang, André Barreto, Will Dabney, Marc G. Bellemare, and Mark Rowland. A distributional analogue to the successor representation. In *International Conference on Machine Learning (ICML)*, 2024.

[31] Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations (ICLR)*, 2018.

[32] Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *International Conference on Machine learning (ICML)*, 2005.

[33] Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(10):2169–2231, 2007.

[34] Kaixin Wang, Kuangqi Zhou, Jiashi Feng, Bryan Hooi, and Xinchao Wang. Reachability-aware Laplacian representation in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

[35] Martin Klissarov and Marlos C. Machado. Deep Laplacian-based options for temporally-extended exploration. In *International Conference on Machine Learning (ICML)*, 2023.

[36] Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G. Bellemare. On the generalization of representations in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[37] Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G. Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2023.

[38] Payam Piray and Nathaniel D Daw. Reconciling flexibility and efficiency: medial entorhinal cortex represents a compositional cognitive map. *Nature communications*, 16(1):7444, 2025.

[39] Siddarth Chandrasekar and Marlos C Machado. Towards an option basis to optimize all rewards. In *Workshop on Reinforcement Learning Beyond Rewards, Reinforcement Learning Conference (RLC)*, 2025.

[40] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

[41] Geoffrey Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude, 2012. URL `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

[42] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL `http://github.com/Farama-Foundation/Gymnasium-Robotics`.

A An Alternative DR Formulation

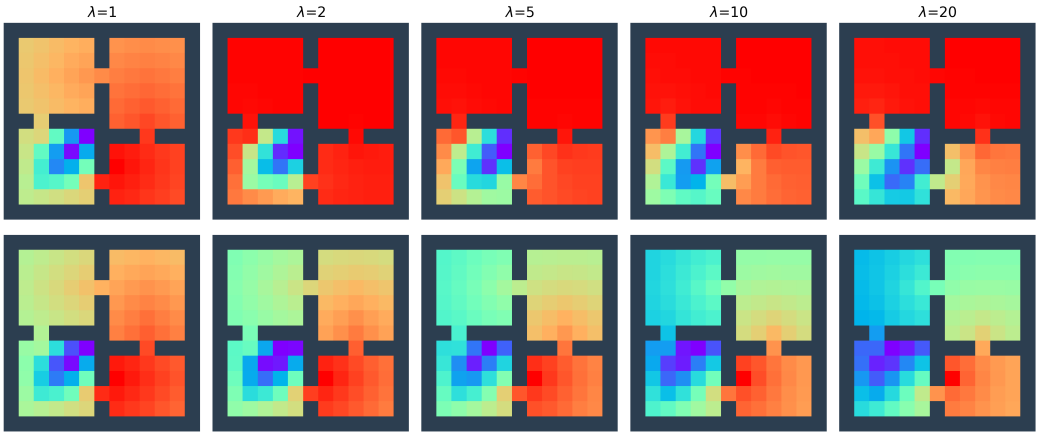

Figure 3: The principal eigenvector of the DR for (top) not treating terminal states as absorbing states [12], and (bottom) treating terminal states as absorbing states in the four rooms environment. See Appendix D.1 for a detailed explanation of the environments.

We formulate the DR slightly differently than prior work [12]. Specifically, we handle a terminal state differently. Let N, T be the set of indices for non-terminal states and terminal states respectively. For example, $\mathbf{P}_{NN}$ denotes the sub-matrix of $\mathbf{P}$ that describes the transition probabilities from non-terminal states to non-terminal states. Recall that the DR is defined as

$$(\mathbf{R} - \mathbf{P})^{-1} = \left(\begin{bmatrix} \text{diag}(\exp(-\mathbf{r}_N/\lambda)) & \mathbf{0} \\ \mathbf{0} & \text{diag}(\exp(-\mathbf{r}_T/\lambda)) \end{bmatrix} - \begin{bmatrix} \mathbf{P}_{NN} & \mathbf{P}_{NT} \\ \mathbf{0} & \mathbf{P}_{TT} \end{bmatrix} \right)^{-1}. \quad (21)$$

Prior work [12] allowed terminal states to emit any rewards at the last time step, but used 0 for most of their experiments, which is standard in RL. When the terminal state reward is 0, if we treat terminal states as absorbing states, i.e., transitioning to themselves with probability 1, then both $\text{diag}(\exp(-\mathbf{r}_T/\lambda))$ and $\mathbf{P}_{TT}$ are equal to $\mathbf{I}$, and the DR is not well defined. To address this issue, Tse et al. [12] considered terminal states as states that transition to a unique absorbing state with probability 1, and they removed this absorbing state from the DR matrix.

While this solution worked well for the experiments Tse et al. [12] considered, where they used a small $\lambda = 1.3$, we find that the DR formulated in this way is not robust to larger values of λ. Specifically, as shown on the top row of Fig. 3, we find that as λ increases, the principal eigenvector loses the property of having the largest value at the terminal state. A consequence is that, when we use such a principal eigenvector in applications such as reward shaping, the shaping reward encourages the agent to visit states other than the terminal states.

We find that we can handle terminal states differently such that we retain this property. Specifically, we treat terminal states as absorbing states, so $\mathbf{P}_{TT} = \mathbf{I}$. Then, to ensure that the DR is well defined, we require terminal states to have a reward of $-\delta$, where $\delta > 0$ is a tiny positive constant. We use $\delta = 0.001$ in our experiments. Using this formulation of the DR, as shown on the bottom row of Fig. 3, the principal eigenvector is the largest at the terminal state even when λ is large.

Note that the theorems proved in [12] can be easily adapted to this new formulation of the DR. Specifically, the symmetrized DR under the new formulation is still guaranteed to have a positive principal eigenvector. Finally, note that such formulations of the terminal states are merely part of the algorithm for computing the DR, and do not alter the underlying MDP. When evaluating the agent, for example, we still treat the environment as an episodic task that terminates with a 0 reward.

B Theory

We provide derivations not included in the main text.

B.1 Proof of Proposition 3.2

Lemma B.1. *$L_{GDO}(\mathbf{u})$ can be written as*

$$L_{GDO}(\mathbf{u}) = \frac{1}{2}\sum_s \sum_{s'} \tilde{\mathbf{P}}(s,s')\big(u(s)-u(s')\big)^2 + \sum_s \exp(-r(s)/\lambda)u(s)^2 + b\left(\sum_s u(s)^2 - c\right)^2. \tag{22}$$

Proof.

$$L_{\text{GDO}}(\mathbf{u}) = \mathbf{u}^\top(\mathbf{I}-\tilde{\mathbf{P}})\mathbf{u} + \mathbf{u}^\top\mathbf{R}\mathbf{u} + b(\mathbf{u}^\top\mathbf{u}-c)^2. \tag{23}$$

It is known that the first term can be written as [14]

$$\mathbf{u}^\top(\mathbf{I}-\tilde{\mathbf{P}})\mathbf{u} = \frac{1}{2}\sum_s\sum_{s'}\tilde{\mathbf{P}}(s,s')\big(u(s)-u(s')\big)^2. \tag{24}$$

Finally, we rewrite $\mathbf{u}^\top\mathbf{R}\mathbf{u}$ and $\mathbf{u}^\top\mathbf{u}$ as summations. □

Proposition 3.2. *Given a transition (s,a,r,s') and an extra state s'', where s and s'' are sampled independently and uniformly at random, and $a\sim\pi_d$. We can minimize $L_{GDO}(\mathbf{u}_{\boldsymbol{\theta}})$ by minimizing*

$$\ell_{GDO}(\boldsymbol{\theta}) = \frac{1}{2}\big(u_{\boldsymbol{\theta}}(s)-u_{\boldsymbol{\theta}}(s')\big)^2 + \exp(-r/\lambda)u_{\boldsymbol{\theta}}(s)^2 + b'(u_{\boldsymbol{\theta}}(s)^2-c')(u_{\boldsymbol{\theta}}(s'')^2-c'), \tag{25}$$

where b' and c' can be treated as hyperparameters.

Proof. Let $n=|\mathcal{S}|$, and ρ be the uniform state distribution. Eq. 22 can be expressed as

$$\frac{1}{2}\sum_s\sum_{s'}\tilde{\mathbf{P}}(s,s')\big(u(s)-u(s')\big)^2 + \sum_s\exp(-r(s)/\lambda)u(s)^2 + b\left(\sum_s u(s)^2-c\right)^2 \tag{26}$$

$$= \frac{n}{2}\sum_s\frac{1}{n}\sum_{s'}\tilde{\mathbf{P}}(s,s')\big(u(s)-u(s')\big)^2 + n\sum_s\frac{1}{n}\exp(-r(s)/\lambda)u(s)^2 + b\left(n\sum_s\frac{1}{n}u(s)^2-c\right)^2 \tag{27}$$

$$= \frac{n}{2}\mathbb{E}_{s\sim\rho}\left[\mathbb{E}_{s'\sim\tilde{\mathbf{P}}(s,\cdot)}\left[\big(u(s)-u(s')\big)^2\right]\right] + n\mathbb{E}_{s\sim\rho}\left[\exp(-r(s)/\lambda)u(s)^2\right] + bn^2\left(\mathbb{E}_{s\sim\rho}[u(s)^2]-c/n\right)^2 \tag{28}$$

$$\propto \frac{1}{2}\mathbb{E}_{s\sim\rho}\left[\mathbb{E}_{s'\sim\tilde{\mathbf{P}}(s,\cdot)}\left[\big(u(s)-u(s')\big)^2\right]\right] + \mathbb{E}_{s\sim\rho}\left[\exp(-r(s)/\lambda)u(s)^2\right] + bn\left(\mathbb{E}_{s\sim\rho}[u(s)^2-c/n]\right)^2 \tag{29}$$

$$= \frac{1}{2}\mathbb{E}_{s\sim\rho}\left[\mathbb{E}_{s'\sim\tilde{\mathbf{P}}(s,\cdot)}\left[\big(u(s)-u(s')\big)^2\right]\right] + \mathbb{E}_{s\sim\rho}\left[\exp(-r(s)/\lambda)u(s)^2\right] + b'\left(\mathbb{E}_{s\sim\rho}[u(s)^2-c']\right)^2. \tag{30}$$

Finally, ℓ_{GDO} is just a sample-based approximation of Eq. 30. Note that in the final sampling, we implicitly approximate $s'\sim\tilde{\mathbf{P}}(s,\cdot)$ with $s'\sim\mathbf{P}(s,\cdot)$, simply because sampling from $\tilde{\mathbf{P}}$ requires sampling backward in time. □

B.2 Proof of Proposition 3.4

Proposition 3.4. *The distance in the $\mathbf{v}$-space is characterized by the Riemannian metric tensor*

$$\mathbf{G}(\mathbf{v}) = \operatorname{diag}(\exp(2\mathbf{v})). \tag{31}$$

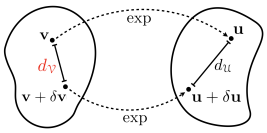

Figure 4: $d_{\mathcal{U}}$ denotes the distance in the $\mathbf{u}$-space, which is assumed to be Euclidean. We aim to find the induced distance in the $\mathbf{v}$-space, denoted by $d_{\mathcal{V}}$, such that $d_{\mathcal{V}}$ between points in the $\mathbf{v}$-space is measured by $d_{\mathcal{U}}$ of their images under the exp transformation.

Proof. Suppose we have an optimization problem

$$\min_{\mathbf{u}\in\mathcal{U}} L(\mathbf{u}), \tag{32}$$

where L is an objective function, and $\mathcal{U}$ is the parameter space for the optimization problem. We assume the Euclidean distance for $\mathcal{U}$. That is, the distance between two points $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{U}$ is computed as $d_{\mathcal{U}}(\mathbf{u}_1, \mathbf{u}_2) = \sqrt{(\mathbf{u}_1 - \mathbf{u}_2)^\top (\mathbf{u}_1 - \mathbf{u}_2)}$. Let $\delta\mathbf{u}$ be a gradient step. The distance between the updated and the original parameters is $d_{\mathcal{U}}(\mathbf{u}, \mathbf{u} + \delta\mathbf{u}) = \sqrt{\delta\mathbf{u}^\top \delta\mathbf{u}} = \|\delta\mathbf{u}\|_2$, which is just the Euclidean norm of the gradient step.

We can re-parameterize and optimize the logarithm of $\mathbf{u}$ instead. We now have the following:

$$\min_{\mathbf{v}\in\mathcal{V}} L(\exp(\mathbf{v})), \tag{33}$$

where exp here denotes element-wise exponentiation, and $\mathcal{V}$ is the parameter space for this new optimization problem.

Let $\delta\mathbf{u}$ be the change induced in $\mathbf{u}$ by $\delta\mathbf{v}$, which is a small change in $\mathbf{v}$. That is, $\delta\mathbf{u} = \exp(\mathbf{v}+\delta\mathbf{v}) - \exp(\mathbf{v})$. Given 1) the relationship $\mathbf{u} = \exp(\mathbf{v})$, and 2) the Euclidean distance for $\mathcal{U}$, the Riemannian metric tensor, $\mathbf{G}(\mathbf{v})$, can be found by solving the following equation for small $\delta\mathbf{v}$ [16]:

$$d^2_{\mathcal{U}}(\mathbf{u}, \mathbf{u} + \delta\mathbf{u}) = d^2_{\mathcal{V}}(\mathbf{v}, \mathbf{v} + \delta\mathbf{v}) = \delta\mathbf{v}^\top \mathbf{G}(\mathbf{v}) \delta\mathbf{v}, \tag{34}$$

where $d_{\mathcal{V}}$ is the corresponding distance in the parameter space $\mathcal{V}$. See Fig. 4 for a visual illustration.

Starting from the left-hand side, we have

$$\begin{aligned}
d^2_{\mathcal{U}}(\mathbf{u}, \mathbf{u} + \delta\mathbf{u}) &= d^2_{\mathcal{U}}(\exp(\mathbf{v}), \exp(\mathbf{v} + \delta\mathbf{v})) && (35)\\
&= \sum_s [\exp(v(s) + \delta v(s)) - \exp(v(s))]^2 && (36)\\
&= \sum_s [\exp(v(s))(\exp(\delta v(s)) - 1)]^2 && (37)\\
&= \sum_s [\exp(v(s))(1 + \delta v(s) + O(\delta v(s)^2) - 1)]^2 \qquad \text{(Taylor expansion)} && (38)\\
&\approx \sum_s [\exp(v(s))\delta v(s)]^2 \qquad (O(\delta v(s)^2) \text{ is ignored since } \delta\mathbf{v} \text{ is small}) && (39)\\
&= \sum_s \exp(2v(s))\delta v(s)^2 && (40)\\
&= \delta\mathbf{v}^\top \operatorname{diag}(\exp(2\mathbf{v}))\delta\mathbf{v}. && (41)
\end{aligned}$$

Therefore, the Riemannian metric tensor is $\mathbf{G}(\mathbf{v}) = \operatorname{diag}(\exp(2\mathbf{v}))$. □

B.3 Proof of Proposition 3.6

Lemma B.4. *The natural gradient of $L^{\log}_{GDO}(\mathbf{v}_{\boldsymbol{\phi}})$ with respect to the output of the neural network parameterized by $\boldsymbol{\phi}$, $\mathbf{v}_{\boldsymbol{\phi}}$, is proportional to*

$$\exp(-\mathbf{r}/\lambda) - \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \tilde{\mathbf{P}}\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top \mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{1}. \tag{42}$$

Proof.

$$\frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}} = \frac{\partial}{\partial \mathbf{u}_{\boldsymbol{\phi}}}(\mathbf{u}_{\boldsymbol{\phi}}^\top(\mathbf{I}+\mathbf{R}-\tilde{\mathbf{P}})\mathbf{u}_{\boldsymbol{\phi}} + b(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c)^2) \tag{43}$$

$$\propto (\mathbf{I}+\mathbf{R}-\tilde{\mathbf{P}})\mathbf{u}_{\boldsymbol{\phi}} + 2b(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c)\mathbf{u}_{\boldsymbol{\phi}} \tag{44}$$

$$= (\mathbf{R}-\tilde{\mathbf{P}})\mathbf{u}_{\boldsymbol{\phi}} + [b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c) + b'/b']\mathbf{u}_{\boldsymbol{\phi}} \tag{45}$$

$$= (\mathbf{R}-\tilde{\mathbf{P}})\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{u}_{\boldsymbol{\phi}}, \tag{46}$$

where we absorb constants into b' and c'. Then, the Hadamard product of $\exp(-\mathbf{v}_{\boldsymbol{\phi}})$ with Eq. 46 is

$$\exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot [(\mathbf{R}-\tilde{\mathbf{P}})\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{u}_{\boldsymbol{\phi}}] \tag{47}$$

$$= \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot (\mathbf{R}-\tilde{\mathbf{P}})\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{1} \tag{48}$$

$$= \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \mathbf{R}\mathbf{u}_{\boldsymbol{\phi}} - \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \tilde{\mathbf{P}}\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{1} \tag{49}$$

$$= \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \exp(-\mathbf{r}/\lambda) \odot \mathbf{u}_{\boldsymbol{\phi}} - \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \tilde{\mathbf{P}}\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{1} \tag{50}$$

$$= \exp(-\mathbf{r}/\lambda) - \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \tilde{\mathbf{P}}\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{1}. \tag{51}$$

□

Lemma B.5. *The s-th entry of Eq. 42 is estimated by*

$$\exp(-r/\lambda) - \exp(v_{\boldsymbol{\phi}}(s') - v_{\boldsymbol{\phi}}(s)) + b''(\exp(2v_{\boldsymbol{\phi}}(s'')) - c''), \tag{52}$$

where (s, a, r, s') is generated by sampling $a \sim \pi_d$, and s'' is a state sampled uniformly at random.

Proof. Let $n = |\mathcal{S}|$, and ρ be the uniform state distribution. The s-th entry of Eq. 42 is

$$[\exp(-\mathbf{r}/\lambda) - \exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \tilde{\mathbf{P}}\mathbf{u}_{\boldsymbol{\phi}} + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c')\mathbf{1}](s) \tag{53}$$

$$= \exp(-r(s)/\lambda) - \exp(-v_{\boldsymbol{\phi}}(s))[\tilde{\mathbf{P}}\mathbf{u}_{\boldsymbol{\phi}}](s) + b'(\mathbf{u}_{\boldsymbol{\phi}}^\top\mathbf{u}_{\boldsymbol{\phi}} - c') \tag{54}$$

$$= \exp(-r(s)/\lambda) - \exp(-v_{\boldsymbol{\phi}}(s))\mathbb{E}_{s'\sim\tilde{\mathbf{P}}(s,\cdot)}[u_{\boldsymbol{\phi}}(s')] + b'\left(\sum_{s''} u_{\boldsymbol{\phi}}^2(s'') - c'\right) \tag{55}$$

$$= \exp(-r(s)/\lambda) - \exp(-v_{\boldsymbol{\phi}}(s))\mathbb{E}_{s'\sim\tilde{\mathbf{P}}(s,\cdot)}[\exp(v_{\boldsymbol{\phi}}(s'))] + b'\left(n\sum_{s''}\frac{1}{n}u_{\boldsymbol{\phi}}^2(s'') - c'\right) \tag{56}$$

$$= \exp(-r(s)/\lambda) - \mathbb{E}_{s'\sim\tilde{\mathbf{P}}(s,\cdot)}[\exp(v_{\boldsymbol{\phi}}(s') - v_{\boldsymbol{\phi}}(s))] + b'n\left(\mathbb{E}_{s''\sim\rho}[u_{\boldsymbol{\phi}}^2(s'')] - c'/n\right) \tag{57}$$

$$= \exp(-r(s)/\lambda) - \mathbb{E}_{s'\sim\tilde{\mathbf{P}}(s,\cdot)}[\exp(v_{\boldsymbol{\phi}}(s') - v_{\boldsymbol{\phi}}(s))] + b''\left(\mathbb{E}_{s''\sim\rho}[\exp(2v_{\boldsymbol{\phi}}(s''))] - c''\right). \tag{58}$$

Eq. 52 is a sample-based approximation of the above equation. □

Proposition 3.6. *Given a transition (s, a, r, s') and an extra state s'', where s and s'' are sampled independently and uniformly at random, and $a \sim \pi_d$. We can minimize $L_{GDO}^{\log}(\mathbf{v}_{\boldsymbol{\phi}})$ with natural gradient descent by minimizing*

$$\ell_{GDO}^{NG}(\boldsymbol{\phi}) = [\![\exp(-r(s)/\lambda) - \exp(v_{\boldsymbol{\phi}}(s') - v_{\boldsymbol{\phi}}(s)) + b''(\exp(2v_{\boldsymbol{\phi}}(s'')) - c'')]\!]v_{\boldsymbol{\phi}}(s), \tag{59}$$

where $[\![\cdot]\!]$ denotes stop-gradient, and λ, b'', c'' are hyperparameters.

Proof. Let $n = |\mathcal{S}|$, and ρ be the uniform state distribution. Recall that the gradient of $L_{\text{GDO}}^{\log}(\mathbf{v}_{\boldsymbol{\phi}})$ with respect to the i-th entry of the parameters, $\boldsymbol{\phi}$, is (see Eq. 11)

$$\frac{\partial \mathbf{v}_{\boldsymbol{\phi}}}{\partial \phi_i} \cdot \left(\exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}}\right) \tag{60}$$

$$= \sum_s \left[\frac{\partial \mathbf{v}_{\boldsymbol{\phi}}}{\partial \phi_i}\right](s) \left[\exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}}\right](s) \tag{61}$$

$$= n\sum_s \frac{1}{n}\frac{\partial v_{\boldsymbol{\phi}}(s)}{\partial \phi_i} \left[\exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}}\right](s) \tag{62}$$

$$\propto \mathbb{E}_{s\sim\rho}\left[\frac{\partial v_{\boldsymbol{\phi}}(s)}{\partial \phi_i} \left[\exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}}\right](s)\right] \tag{63}$$

We can estimate this expectation by sampling a state, s, uniformly at random. Applying Lemmas B.4 and B.5 gives

$$\frac{\partial v_{\boldsymbol{\phi}}(s)}{\partial \phi_i}\left[\exp(-\mathbf{v}_{\boldsymbol{\phi}}) \odot \frac{\partial L_{\text{GDO}}(\mathbf{u}_{\boldsymbol{\phi}})}{\partial \mathbf{u}_{\boldsymbol{\phi}}}\right](s) \tag{64}$$

$$\propto \frac{\partial v_{\boldsymbol{\phi}}(s)}{\partial \phi_i}\big(\exp(-r/\lambda) - \exp(v_{\boldsymbol{\phi}}(s') - v_{\boldsymbol{\phi}}(s)) + b''(\exp(2v_{\boldsymbol{\phi}}(s'')) - c'')\big), \tag{65}$$

where (s, a, r, s') is the transition generated by sampling $a \sim \pi_d$, and s'' is another state sampled uniformly at random. Eq. 65 is a sample-based approximation of the gradient of $L^{\log}_{\text{GDO}}(\mathbf{v}_{\boldsymbol{\phi}})$ with respect to ϕ_i. Finally, it is trivial to show that the gradient of $\ell^{\text{NG}}_{\text{GDO}}(\boldsymbol{\phi})$ recovers Eq. 65. □

B.4 Extending Proposition 3.7 to Multiple Terminal States

We extend the theoretical analysis in Proposition 3.7 to cover the case of multiple terminal states.

Suppose that we use all terminal states as anchors, while anchoring their values to $\log \mathbf{w}$, i.e., the i-th terminal state is anchored to value $\log w_i$. Here, w_i can be thought of as a preference weight for the i-th terminal state, with more preferred states having larger values. If there is no preference among terminal states, we can simply set $\mathbf{w} = \mathbf{1}$. Under this setting, we can show that the solution to the optimization problem captures the aggregated reward-aware potential from all states to terminal states, weighted by their preference weights.

Proof. We now solve the optimization problem

$$\min_{\mathbf{v}} \exp(\mathbf{v})^\top (\mathbf{R} - \tilde{\mathbf{P}}) \exp(\mathbf{v}) \quad \text{s.t.} \quad v(s_j) = \log w_j \; \forall s_j \in \mathcal{S}_T. \tag{66}$$

Following similar steps as Eq. 14 to 17, we know that the solution to the optimization problem is

$$\exp(\mathbf{v}) = \sum_j \frac{-\mu_j}{2w_j} (\mathbf{R} - \tilde{\mathbf{P}})^{-1} \mathbf{b}_{\mathbf{s_j}}, \tag{67}$$

where μ_j are the Lagrange multipliers, and $\mathbf{b}_{s_j}$ is the basis vector activated at s_j. We can see that the solution is a linear combination of all terminal-state columns of the symmetrized DR, with weights $\frac{-\mu_j}{2w_j}$. This can be written in the more compact form $\tilde{\mathbf{Z}}_{:,T}\mathbf{c}_T$, where $\tilde{\mathbf{Z}}_{:,T}$ are the terminal state columns, and $\mathbf{c}_T$ are the weights that combine them.

We further characterize $\mathbf{c}_T$. When the anchoring constraints are satisfied, we know $\tilde{\mathbf{Z}}_{TT}\mathbf{c}_T = \mathbf{w}$, and therefore $\mathbf{c}_T = \tilde{\mathbf{Z}}_{TT}^{-1}\mathbf{w}$. The solution of the optimization problem can then be written as $\tilde{\mathbf{Z}}_{:,T}\tilde{\mathbf{Z}}_{TT}^{-1}\mathbf{w}$.

Let N, T denote the set of indices for non-terminal and terminal states respectively. Applying block matrix inversion, the solution's expression simplifies to

$$\begin{bmatrix} \frac{1}{2}(\mathbf{R}_{\text{NN}} - \frac{\mathbf{P}_{\text{NN}} + \mathbf{P}_{\text{NN}}^\top}{2})^{-1}\mathbf{P}_{\text{NT}}\mathbf{w} \\ \mathbf{w} \end{bmatrix}. \tag{68}$$

As discussed by Tse et al. [12] (Appendix B.1.1), $(\mathbf{R}_{\text{NN}} - \frac{\mathbf{P}_{\text{NN}} + \mathbf{P}_{\text{NN}}^\top}{2})^{-1}$ is simply the reward-aware distance among non-terminal states under the symmetrized transition dynamics, while $\mathbf{P}_{\text{NT}}$ is the one-step transition probabilities from non-terminal states to terminal states. Therefore, the solution vector simply captures the reward-aware distances from all non-terminal states to terminal states, weighted by their preference weights $\mathbf{w}$.

Simply put, when there are multiple terminal states with preference weights $\mathbf{w}$, DROGO learns an aggregated potential function that measures how close each state is to the preferred terminal states. □

B.5 Proof of Theorem 3.8

Theorem 3.8. *Let $r(s_T) = -\delta \; \forall s_T \in \mathcal{S}_T$, where $\delta > 0$. When $\delta \to 0^+$, the columns of the DR ($(\mathbf{R} - \mathbf{P})^{-1}$) corresponding to all terminal states form a basis for the eigenspace of the principal eigenvalue of the DR.*

Proof. Recall that the (non-symmetrized) DR is $(\mathbf{R}-\mathbf{P})^{-1} = (\text{diag}(\exp(-\mathbf{r}/\lambda)) - \mathbf{P})^{-1}$. We assume without loss of generality that $\lambda = 1$. Let $n = |\mathcal{S}|$ and $m = |\mathcal{S}_T|$. Suppose we order the states such that the m terminal states are ordered last, i.e., they correspond to the last columns of the DR. Then, $\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$, for some matrices $\mathbf{A}$ and $\mathbf{B}$, and $\mathbf{R}-\mathbf{P} = \begin{bmatrix} \mathbf{A}' & -\mathbf{B} \\ \mathbf{0} & (\exp(\delta)-1)\mathbf{I} \end{bmatrix}$ for some matrix $\mathbf{A}'$. When $\delta \to 0^+$, $\exp(\delta) - 1 \approx \delta$, so we write $\mathbf{R}-\mathbf{P}$ as $\begin{bmatrix} \mathbf{A}' & -\mathbf{B} \\ \mathbf{0} & \delta\mathbf{I} \end{bmatrix}$.

It is easy to see that δ is an eigenvalue of $\mathbf{R}-\mathbf{P}$ with algebraic multiplicity m. Note that $\mathbf{R}-\mathbf{P}$ might have complex eigenvalues, and we order eigenvalues by their magnitudes. Using the Gershgorin circle theorem, we can show that all eigenvalues of $\mathbf{R}-\mathbf{P}$ have non-zero magnitude. It is obvious that for the rows corresponding to terminal states, the Gershgorin discs are centered at δ with radius 0. For a row i corresponding to a non-terminal state, the Gershgorin disc is centered at $\exp(-r(s_i)) - P(i,i)$. Since we assume $r(s) < 0\ \forall s \in \mathcal{S}$, $\exp(-r(s_i)) - P(i,i) > 1 - P(i,i) = \sum_{j \neq i} P(i,j)$, where $\sum_{j\neq i} P(i,j)$ is the radius of the disc. Thus, any discs corresponding to non-terminal states do not contain 0. Then, all eigenvalues of $\mathbf{R}-\mathbf{P}$ must have magnitudes greater than 0. Since δ can be arbitrarily small, δ is the eigenvalue of $\mathbf{R}-\mathbf{P}$ with the smallest magnitude, and thus the principal eigenvalue of the DR, $(\mathbf{R}-\mathbf{P})^{-1}$.

We now study the principal eigenvectors of the DR, which are equal to the eigenvectors of $\mathbf{R}-\mathbf{P}$ corresponding to δ. An eigenvector corresponding to δ satisfies

$$\begin{bmatrix} \mathbf{A}' & -\mathbf{B} \\ \mathbf{0} & \delta\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \delta \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, \tag{69}$$

which gives the equation

$$\mathbf{A}'\mathbf{x} - \mathbf{B}\mathbf{y} = \delta\mathbf{x} \tag{70}$$

$$\mathbf{x} = (\mathbf{A}' - \delta\mathbf{I})^{-1}\mathbf{B}\mathbf{y} \tag{71}$$

An eigenvector corresponding to δ then has the general form

$$\begin{bmatrix} (\mathbf{A}' - \delta\mathbf{I})^{-1}\mathbf{B}\mathbf{y} \\ \mathbf{y} \end{bmatrix}, \tag{72}$$

where $\mathbf{y}$ can be any vector in $\mathbb{R}^m$. Therefore, the geometric multiplicity of δ is also m.

We now derive the form of the columns of the DR corresponding to terminal states. By block-wise matrix inversion, the DR is

$$\begin{bmatrix} \mathbf{A}' & -\mathbf{B} \\ \mathbf{0} & \delta\mathbf{I} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A}')^{-1} & \frac{1}{\delta}(\mathbf{A}')^{-1}\mathbf{B} \\ \mathbf{0} & \frac{1}{\delta}\mathbf{I} \end{bmatrix}. \tag{73}$$

Then, the columns corresponding to the terminal states have the form

$$\begin{bmatrix} \frac{1}{\delta}(\mathbf{A}')^{-1}\mathbf{B}\mathbf{b}_j \\ \frac{1}{\delta}\mathbf{b}_j \end{bmatrix}, \tag{74}$$

where $\mathbf{b}_i \in \mathbb{R}^m$ is the standard basis vector with a 1 at entry i, and $j \in \{1, 2, \ldots, m\}$. Comparing Eq. 72 and 74, we can see that as $\delta \to 0^+$, these linearly independent columns of the DR span the same space as the eigenvectors corresponding to the eigenvalue δ. Thus, they form a basis for the eigenspace of the principal eigenvalue of the DR. □

B.6 Proof of Theorem 3.10

Theorem 3.10. *When there is a single terminal state, the angle between the terminal-state column and the principal eigenvector of the symmetrized DR, θ, satisfies*

$$\tan\theta \leq \frac{|\mu_1|}{|\mu_2||w_1(s_T)|}, \tag{75}$$

where μ_1, μ_2 are the smallest and second smallest eigenvalues of $\mathbf{R}-\tilde{\mathbf{P}}$ respectively, and $w_1(s_T)$ is the entry of the principal eigenvector of the symmetrized DR corresponding to the terminal state.

Proof. We first focus on the case with a single terminal state. Let $\mu_1, \mu_2, \dots$ and $\mathbf{w_1}, \mathbf{w}_2, \dots$ be the eigenvalues and eigenvectors of $(\mathbf{R} - \tilde{\mathbf{P}})$ respectively, where the eigenvalues are ordered by their magnitudes ascendingly. Since $(\mathbf{R} - \tilde{\mathbf{P}})$ is symmetric, we know that the eigenvectors can be chosen to be orthonormal. A terminal-state column of the symmetrized DR can be written as

$$\sum_i \frac{1}{\mu_i} \mathbf{w}_i \mathbf{w}_i^\top \mathbf{b}_{s_T} = \sum_i \frac{w_i(s_T)}{\mu_i} \mathbf{w}_i. \tag{76}$$

We can see that the terminal-state column can be written as the sum of two components: (1) the component along the direction of the principal eigenvector, $\frac{w_1(s_T)}{\mu_1}\mathbf{w}_1$, and (2) the component orthogonal to the principal eigenvector, $\sum_{i\geq 2} \frac{w_i(s_T)}{\mu_i}\mathbf{w}_i$. The norm of the first component is simply $\frac{|w_1(s_T)|}{|\mu_1|}$, while that of the second component is

$$\left\| \sum_{i\geq 2} \frac{w_i(s_T)}{\mu_i} \mathbf{w}_i \right\|_2^2 = \sum_{i\geq 2} \frac{w_i(s_T)^2}{\mu_i^2} \leq \frac{1}{\mu_2^2} \sum_{i\geq 2} w_i(s_T)^2 \leq \frac{1}{\mu_2^2}. \tag{77}$$

The angle between the terminal-state column and the principal eigenvector then satisfies

$$\tan\theta = \frac{\| \sum_{i\geq 2} \frac{w_i(s_T)}{\mu_i} \mathbf{w}_i \|_2}{\| \frac{w_1(s_T)}{\mu_1} \mathbf{w}_1 \|_2} \leq \frac{|\mu_1|}{|\mu_2||w_1(s_T)|}. \tag{78}$$

Note that this bound takes into account the effect of both symmetrization and δ, as δ affects μ_1, μ_2 and $\mathbf{w}_1$. We note that it is possible to repeat the above derivation for the case of multiple terminal states. When there are multiple terminal states, for any terminal state s_T, the angle between the corresponding column and the principal eigenspace satisfies

$$\tan\theta \leq \frac{|\mu_{|\mathcal{S}_T|}|}{|\mu_{|\mathcal{S}_T|+1}| \|\mathbf{W}^\top \mathbf{b}_{s_T}\|_2}, \tag{79}$$

where $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_{|\mathcal{S}_\mathbf{T}|}]$. □

C Effect of a Finite δ and Symmetrization on Approximating the Principal Eigenvector

We empirically measure the discrepancy between the terminal-state column and the principal eigenvector for the four environments considered in our paper for both the non-symmetrized and symmetrized DR cases.

Table 3: The cosine similarity between the terminal-state column and the principal eigenvector for different non-zero finite δ for the non-symmetrized DR.

δ	1.0	0.1	0.01	0.001	0.0001
GRID TASK	0.916	0.999	1.000	1.000	1.000
FOUR ROOMS	0.658	0.999	1.000	1.000	1.000
GRID MAZE	0.662	0.999	1.000	1.000	1.000
GRID ROOM	0.504	0.999	1.000	1.000	1.000

Theorem 3.8 states that as $\delta \to 0$, the terminal-state column spans the principal eigenspace. From the above table, we can see that even when using a finite non-zero δ as large as 0.1, the gap between the terminal-state column and the principal eigenvector is extremely small, so the effect of using a finite δ is minimal in the environments we consider.

For the symmetrized DR, we report the lower bound of the cosine similarity (inferred from our $\tan\theta$ bound in Theorem 3.10) and the actual cosine similarity between the terminal-state column and the principal eigenvector. We can see that for a large range of δ (0.1 to 0.0001), our bound predicts that the cosine similarity is at least around 0.95, while we see that the actual cosine similarity is around 0.99.

Table 4: The cosine similarity (predicted lower bound/actual value) between the terminal-state column and the principal eigenvector for different non-zero finite δ for the symmetrized DR.

δ	1.0	0.1	0.01	0.001	0.0001
GRID TASK	0.896/0.995	0.956/0.989	0.949/0.988	0.949/0.988	0.949/0.988
FOUR ROOMS	0.785/0.989	0.968/0.993	0.962/0.992	0.961/0.992	0.961/0.992
GRID MAZE	0.727/0.975	0.994/0.999	0.991/0.999	0.991/0.999	0.991/0.999
GRID ROOM	0.753/0.986	0.983/0.997	0.977/0.996	0.977/0.996	0.976/0.996

From both our theoretical and empirical results, we can see that even though symmetrization leads to degradation and plateauing of the cosine similarity between the terminal-state column and the principal eigenvector, the degradation is minimal, and the terminal-state column remains a close approximation of the principal eigenvector.

D Experiment Details

We present more experiment details. Experiments are run using NVIDIA L40S GPUs. Each run takes approximately one hour for one-hot and coordinates inputs, and roughly 4 hours for pixels inputs.

D.1 Environments

We use the set of environments shown in Fig. 5.

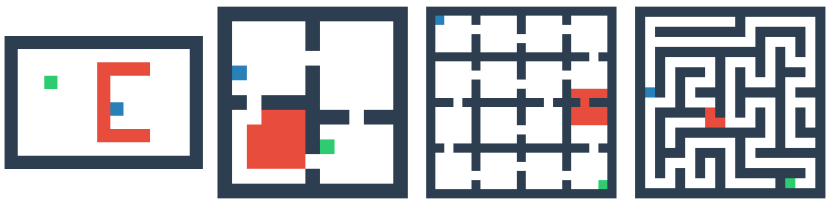

Figure 5: The set of episodic environments used in [12]. From left to right: 1) grid task, 2) four rooms, 3) grid room, and 4) grid maze. The start state is in blue. The agent receives -1 reward at every time step unless it steps on red tiles (-20 reward) or reaches the goal in green (0 reward).

D.2 Details on DROGO Experiments

We provide more details on using DROGO to approximate the principal eigenvector of the DR.

D.2.1 Neural Network Architecture

We use the same neural network architecture for one-hot and coordinates representations of states. The network consists of 4 fully connected layers with 128 hidden units followed by a final output layer. For pixel inputs, the network consists of 3 convolutional layers (16, 32, and 64 filters respectively) with 3×3 kernels, stride 1, and padding of 1. The first two convolutional layers are each followed by 2×2 max-pooling with stride 2. The output is then flattened and fed to a fully connected layer with 128 hidden units, followed by a final output layer. All fully connected and convolutional layers, except for the output layer, are followed by the rectified linear unit (ReLU) activation [40].

D.2.2 Hyperparameters

We tune the hyperparameters manually. We use the same set of hyperparameters for all environments and all state representations. We use $\lambda = 20$, which prevents the $\exp(-r(s)/\lambda)$ term in the DROGO loss (Eq. 19) from being too large and causing instabilities. We use the RMSprop optimizer [41], a step size of $1e{-5}$, and a batch size of 2000. We apply gradient norm clipping of 0.5. For coordinates and pixel inputs, we normalize the inputs to have the range $[-0.5, 0.5]$.

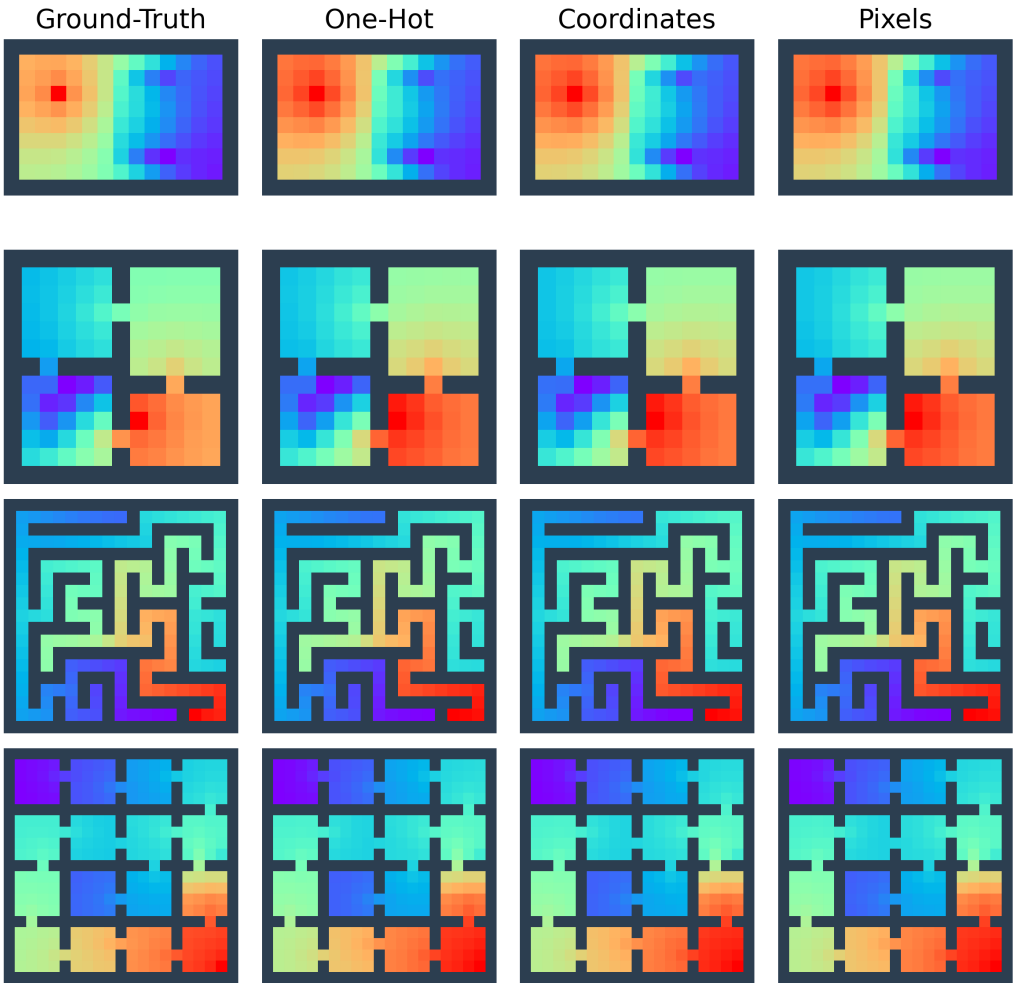

Figure 6: Visualization of the vectors learned using DROGO, averaged over 10 seeds, for different state representations. Columns from left to right: 1) the ground-truth principal eigenvectors of the DR, 2) the vectors learned for one-hot inputs, 3) the vectors learned for coordinates inputs, and 4) the vectors learned for pixels inputs. Note that the eigenvectors here appear visually different from those presented by Tse et al. [12] since they used $\lambda = 1.3$, while we use $\lambda = 20$.

D.2.3 Visualization of the Learned and Ground-truth Principal Eigenvectors

We visually compare the learned vectors and the ground-truth principal eigenvectors of the DR in Fig. 6.

D.3 Sensitivity of DROGO to λ

While we use $\lambda = 20$ for DROGO in our experiments, we also perform a sensitivity analysis of DROGO with respect to λ. Specifically, we consider $\lambda \in \{2, 5, 10, 20, 40, 80\}$. We present in Table 5 the cosine similarity between the approximated vector and the ground-truth principal eigenvector of the DR for different λ, and present in Table 6 and 7 the effect of λ on N_{OPT} and N_{VISIT} in reward shaping respectively. As we can observe from the tables, DROGO is rather robust to the choice of λ.

Table 5: Sensitivity of the cosine similarity to λ.

λ	2	5	10	20	40	80
GRID TASK	0.511	0.985	0.998	0.996	0.992	0.986
FOUR ROOMS	0.619	0.999	0.999	0.998	0.995	0.991
GRID MAZE	-	0.900	0.968	1.000	1.000	0.999
GRID ROOM	0.823	0.960	0.995	1.000	0.999	0.999

Table 6: Sensitivity of N_{OPT} in reward shaping to λ.

λ	2	5	10	20	40	80
GRID TASK	$3.14^{+0.32}_{-0.38}$	$1.11^{+0.20}_{-0.14}$	$0.80^{+0.04}_{-0.04}$	$0.98^{+0.04}_{-0.05}$	$1.00^{+0.04}_{-0.04}$	$1.03^{+0.05}_{-0.05}$
FOUR ROOMS	$9.46^{+9.56}_{-7.18}$	$0.16^{+0.00}_{-0.00}$	$0.16^{+0.00}_{-0.00}$	$0.16^{+0.00}_{-0.00}$	$1.92^{+1.00}_{-0.90}$	$3.80^{+5.50}_{-3.05}$
GRID MAZE	-	$21.34^{+3.29}_{-2.28}$	$12.90^{+3.57}_{-3.69}$	$6.46^{+1.14}_{-0.72}$	$6.72^{+1.36}_{-0.94}$	$6.62^{+1.20}_{-0.68}$
GRID ROOM	$42.90^{+0.94}_{-1.48}$	$19.13^{+1.03}_{-1.08}$	$9.77^{+3.29}_{-2.38}$	$7.72^{+0.25}_{-0.32}$	$7.72^{+1.16}_{-0.67}$	$7.31^{+1.26}_{-0.73}$

Table 7: Sensitivity of N_{VISIT} in reward shaping to λ.

λ	2	5	10	20	40	80
GRID TASK	$6.70^{+0.36}_{-0.36}$	$6.91^{+0.84}_{-0.90}$	$6.44^{+0.63}_{-0.54}$	$6.61^{+0.80}_{-0.77}$	$6.54^{+0.41}_{-0.35}$	$6.89^{+0.70}_{-0.67}$
FOUR ROOMS	$7.42^{+13.42}_{-6.96}$	$0.04^{+0.03}_{-0.03}$	$0.02^{+0.03}_{-0.02}$	$0.07^{+0.03}_{-0.03}$	$0.04^{+0.03}_{-0.03}$	$0.03^{+0.03}_{-0.02}$
GRID MAZE	-	$0.38^{+0.05}_{-0.05}$	$0.42^{+0.04}_{-0.04}$	$0.46^{+0.03}_{-0.03}$	$0.47^{+0.04}_{-0.04}$	$0.46^{+0.04}_{-0.04}$
GRID ROOM	$0.86^{+0.04}_{-0.04}$	$0.45^{+0.02}_{-0.02}$	$0.51^{+0.07}_{-0.05}$	$0.55^{+0.05}_{-0.04}$	$0.62^{+0.06}_{-0.06}$	$0.65^{+0.06}_{-0.06}$

D.4 Comparison of Objectives

We empirically compare ℓ_{DROGO} (Eq. 19), $\ell^{\text{NG}}_{\text{GDO}}$ (Eq. 12), $\ell^{\log}_{\text{GDO}}$ (Eq. 8), and ℓ_{GDO} (Eq. 6). Recall that only ℓ_{GDO} does not perform log-space parameterization. Of the remaining three, $\ell^{\log}_{\text{GDO}}$ does not adopt natural gradient. Finally, ℓ_{DROGO} uses the point-wise constraint, while $\ell^{\text{NG}}_{\text{GDO}}$ uses the quadratic norm penalty. See Table 8 for a summary.

To compare these loss functions, we compute the cosine similarity between the vectors learned using the loss functions and the logarithm of the ground-truth principal eigenvector of the DR. Note that for ℓ_{GDO}, there is no guarantee that we can safely apply the logarithm to the learned vector, so we apply the logarithm to the absolute values of the learned vector.

We first compare these loss functions in the tabular case, where we represent the learned vector using an array of length $|\mathcal{S}|$. Let b be the hyperparameter determining the strength of the quadratic penalty term, and α be the step size. For ℓ_{GDO}, $\ell^{\log}_{\text{GDO}}$, and $\ell^{\text{NG}}_{\text{GDO}}$, we perform a grid search over $\alpha \in [10^{-5}, 3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}]$ and $b \in [0.1, 0.5, 1, 2]$. For ℓ_{DROGO}, which does not require b, we simply perform a grid search over the same values of α. Note that we perform the hyperparameter search in the largest environment, grid room, and use the best hyperparameters found for all environments.

Fig. 7 shows the cosine similarity between the learned and ground-truth principal eigenvectors of the DR for the different loss functions. Apart from $\ell^{\log}_{\text{GDO}}$, all loss functions learn vectors that converge to the ground-truth principal eigenvector. Note that $\ell^{\text{NG}}_{\text{GDO}}$ appears to have poor performance in grid room since one out of 10 seeds fails to converge. ℓ_{GDO}, which simply applies the GDO without log-space parameterization, performs well in the tabular setting. However, as we will see later, this loss function does not work well with neural networks. For the remaining loss functions, $\ell^{\log}_{\text{GDO}}$ fails to converge to the ground-truth eigenvector, while $\ell^{\text{NG}}_{\text{GDO}}$ and ℓ_{DROGO} do, confirming the importance of the natural gradient.

We now compare the loss functions using neural networks taking the coordinates of states as inputs. For ℓ_{DROGO}, we tune the hyperparameters manually, as detailed in Appendix D.2. Because we tuned more hyperparameters, we considered fewer values per hyperparameter, effectively considering fewer hyperparameter settings than the grid search described above. For the remaining loss functions, we use

Table 8: The characterization of different loss functions.

Loss Function	Log-Space Parameterization	Natural Gradient	Quadratic Norm Penalty
ℓ_{GDO}	✗	✗	✓
$\ell^{\log}_{\text{GDO}}$	✓	✗	✓
$\ell^{\text{NG}}_{\text{GDO}}$	✓	✓	✓
ℓ_{DROGO}	✓	✓	✗

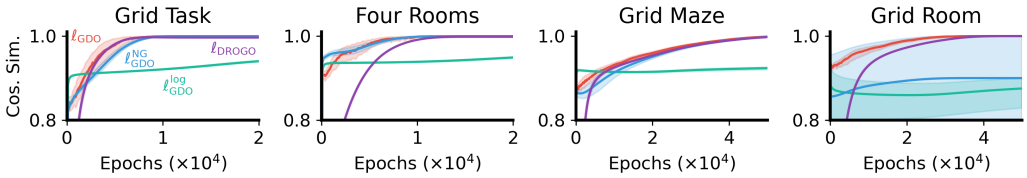

Figure 7: The cosine similarity, averaged over 10 seeds, between the learned and ground-truth principal eigenvectors for different loss functions in the tabular setting. The shaded area indicates 95% bootstrap confidence interval.

the same hyperparameters as ℓ_{DROGO} except for the step size, α, and the quadratic penalty coefficient, b. We identify the values for α and b using the same search procedure as in the tabular case.

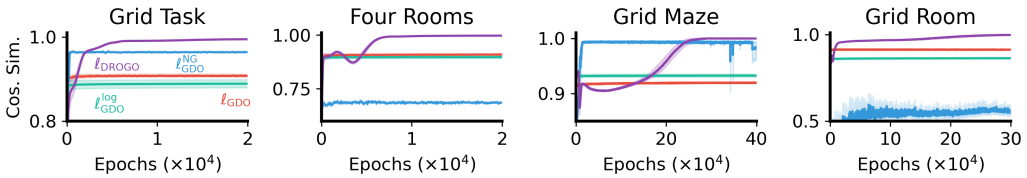

Figure 8: The cosine similarity, averaged over 10 seeds, between the learned and ground-truth principal eigenvectors for different loss functions when using coordinates as inputs to the neural network. The shaded area indicates 95% bootstrap confidence interval.

Figure 8 shows the cosine similarity between the learned and ground-truth principal eigenvectors of the DR. We can see that only ℓ_{DROGO} can learn the principal eigenvector. While in the tabular setting, we observe that vectors learned using ℓ_{GDO} converge to the ground-truth principal eigenvector, the approximation errors with neural networks and the small magnitudes of the ground-truth eigenvector prevent it from learning the principal eigenvector well in the function approximation case. We empirically observe that the vectors learned using ℓ_{GDO} have negative values. $\ell^{\log}_{\text{GDO}}$, on the other hand, fails for the same reason of lacking natural gradient as in the tabular setting. Finally, using the quadratic norm penalty ($\ell^{\text{NG}}_{\text{GDO}}$) fails due to the difficulty of balancing b. When b is large, the penalty interferes with the GDO. When b is small, it is not sufficient to constrain the norm of the learned vector.

D.5 Sensitivity of $\ell^{\text{NG}}_{\text{GDO}}$ to b''

As described in Section 3.3, we observe that when we optimize $\ell^{\text{NG}}_{\text{GDO}}$ in practice, the optimization is very sensitive to the hyperparameter b'', causing failures to approximate the principal eigenvector. Recall that b'' is the hyperparameter balancing the GDO, $\exp(\mathbf{v})^\top(\mathbf{I} + \mathbf{R} - \tilde{\mathbf{P}})\exp(\mathbf{v})$, and the quadratic norm penalty, $(\exp(\mathbf{v})^\top \exp(\mathbf{v}) - c)^2$. We now demonstrate this over-sensitivity.

We use coordinates as inputs to the neural network, and focus on the largest environment, grid room. For $\ell^{\text{NG}}_{\text{GDO}}$ (see Eq. 12), we choose $c'' = 1$, which encourages $\exp(\mathbf{v})^\top \exp(\mathbf{v}) \approx |\mathcal{S}|$. We use the neural network architecture outlined in Appendix D.2.1, and hyperparameters described in

Appendix D.2.2, except for the step size. We investigate $b'' \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2\}$, and for each b'', we perform a grid search over the step size ($\{10^{-5}, 3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}\}$).

Fig. 9 shows the results when optimizing $\ell^{\text{NG}}_{\text{GDO}}$ for different b''. We can see that when b'' is large (0.5, 1, 2), the quadratic norm penalty is successful in constraining the norm of the learned vector, making it stay near $|\mathcal{S}|$. However, the quadratic norm penalty dominates the learning signal, and the cosine similarity between the learned and ground-truth eigenvectors is low. As b'' decreases, the quadratic norm penalty has less importance, allowing GDO to dominate the learning signal, explaining the increased cosine similarity. However, the quadratic norm penalty is no longer effective in constraining the norm of the learned vector, and when b'' is too small (0.01, 0.02, 0.05), the norm of the learned vector first collapses to near 0, before exploding. Because of the instability in the norm, the cosine similarity for these b'' stays near 0.8, and cannot converge to near 1.

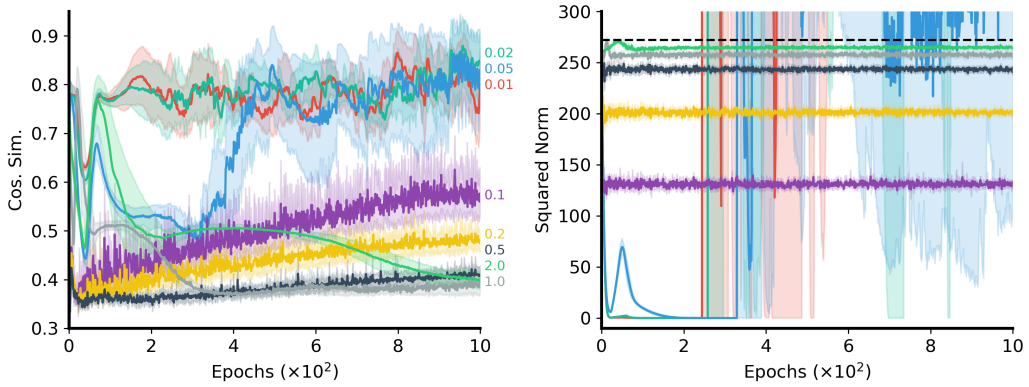

Figure 9: *Left:* The cosine similarity between the learned vectors and the ground-truth eigenvectors for different values of b''. *Right:* The squared norm $(\exp(\mathbf{v})^\top \exp(\mathbf{v}))$ for different b''. The black dashed horizontal line indicates $|\mathcal{S}|$. Shaded areas indicate 95% bootstrapped confidence intervals.

D.6 Learning Curves for Reward Shaping Experiments

We present the learning curves for the reward shaping experiments in Fig. 10. Since the learning curves for different state representations of the DR highly overlap, we here present only the learning curve for the pixels representation of states.

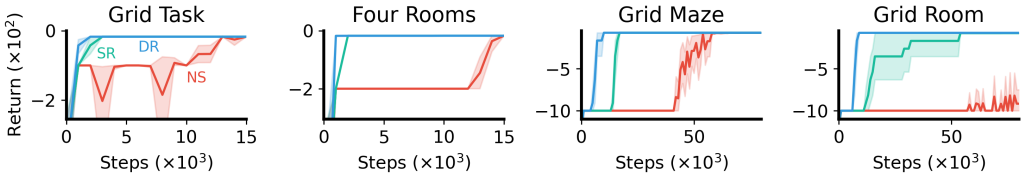

Figure 10: The total return, averaged over 10 seeds, obtained when using the DR for reward shaping, the SR for reward shaping, and not performing reward shaping (NS). The shaded area indicates 95% bootstrap confidence interval.

E DROGO Applied to Continuous State Spaces

To demonstrate the applicability of DROGO to continuous state spaces, we consider three Point Maze environments (see Fig. 11 top row) from the Gymnasium Robotics suite [42]. In these environments, the agent's state is described by 2 real numbers encoding its position, and 2 other real numbers encoding its velocity in two directions. The state space is thus continuous, and contains an uncountably infinite number of states, rendering eigendecomposition impossible.

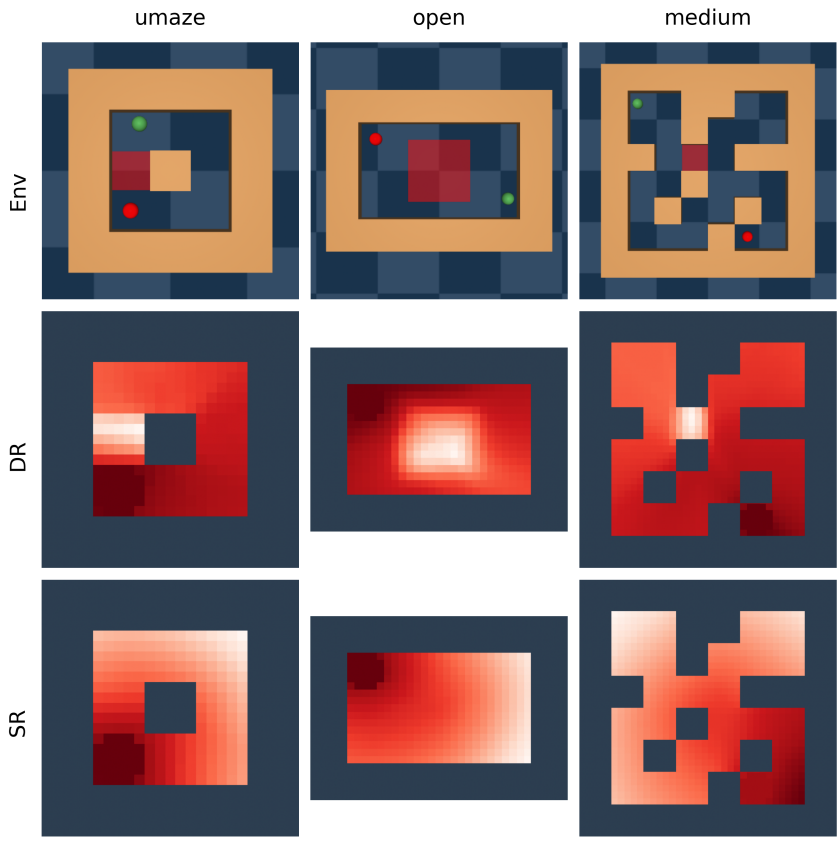

Figure 11: *Top row:* Three Point Maze environments from the Gymnasium Robotics suite [42], adapted to include low-reward regions. Green indicates agent position, red circles indicates goal positions, while red squares indicates low-reward regions. Similar to our grid world experiments, the agent receives -1 reward unless it visits low-reward regions (-20 reward), until it reaches the goal. A key distinction between these environments and earlier grid world environments is that the state space is now continuous, and it is not possible to compute the eigenvectors using eigendecomposition. *Middle row:* the principal eigenvector of the DR approximated using DROGO. *Bottom row:* The principal eigenvector of the SR approximated using ALLO [20].

We apply DROGO to these environments, following a similar procedure as for the grid world environments. We generate 200,000 transitions using a uniform random initial state distribution and the default policy, which we choose to be the uniform random policy. We use the same neural network architecture as detailed in Appendix D.2.1. We use $\lambda = 20$, the RMSprop optimizer [41], a step size of 3×10^{-5}, and a batch size of 20K. We apply gradient norm clipping of 0.5, and normalize all dimensions of the state to have the range $[-0.5, 0.5]$.

Fig. 11 shows the approximated principal eigenvectors of the DR (middle row) and the SR (bottom row) using DROGO and ALLO [20] respectively. Note that we are not able to provide a quantitative measure of how well these learned vectors approximate the ground-truth, similar to that in Fig. 2, precisely because it is not possible to obtain the ground-truth eigenvectors using eigendecomposition. Nevertheless, from the visualization that we provide, we can observe that in these continuous environments, the learned eigenvectors capture the environment dynamics very well, in a way similar to that observed in our previous grid worlds: (1) They assign the highest values to the goal region. (2) As the distance from the goal increases, the values decrease in a topology-aware manner. (3) Near low-reward regions, there is a large decrease in values, demonstrating its reward-awareness.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main claims are supported by evidence in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed limitations in the Conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Assumptions are stated explicitly, and full proofs are provided in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details on model architecture, hyperparameters, and hyperparameter tuning and training procedures in the Appendix. We also submit our code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We submit our code alongside our paper, and provide instructions to generate the results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Details are provided both in the Experiments section and the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report 95% bootstrapped confidence intervals for our plots and tables.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details on compute in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper conforms with the Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper's goal is to advance the field of reinforcement learning. There are potential societal consequences of our work, though none of which we feel must be specifically highlighted.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite original papers and assets whenever possible.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are only used for editing, formatting, and plotting.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.