PROTO SUCCESSOR MEASURE: REPRESENTING THE SPACE OF ALL POSSIBLE SOLUTIONS OF REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Having explored an environment, intelligent agents should be able to transfer their knowledge to most downstream tasks within that environment. Referred to as "zero-shot learning," this ability remains elusive for general-purpose reinforcement learning algorithms. While recent works have attempted to produce zero-shot RL agents, they make assumptions about the nature of the tasks or the structure of the MDP. We present *Proto Successor Measure*: the basis set for all possible solutions of Reinforcement Learning in a dynamical system. We provably show that any possible policy can be represented using an affine combination of these policy independent basis functions. Given a reward function at test time, we simply need to find the right set of linear weights to combine these basis corresponding to the optimal policy. We derive a practical algorithm to learn these basis functions using only interaction data from the environment and show that our approach can produce the optimal policy at test time for any given reward function without additional environmental interactions.

1 INTRODUCTION

A wide variety of tasks can be defined within an environment (or any dynamical system). For instance, in navigation environments, tasks can be defined to reach a goal, path following, reach a goal while avoiding certain states etc. Once familiar with an environment, humans have the wonderful ability to perform new tasks in that environment without any additional practice. For example, consider the last time you moved to a new city. At first, you may have needed to explore various routes to figure out the most efficient way to get to the nearest supermarket or place of work. But eventually, you could probably travel to new places efficiently the very first time you needed to get there. Like humans, intelligent agents should be able to infer the necessary information about the environment during exploration and use this experience for solving any downstream task efficiently. Reinforcement Learning (RL) algorithms have seen great success in finding a sequence of decisions that optimally solves a given task in the environment (Wurman et al., 2022; Fawzi et al., 2022). In RL settings, tasks are defined using reward functions with different tasks having their own optimal agent policy or behavior corresponding to the task reward. RL agents are usually trained for a given task (reward function) or on a distribution of related tasks; most RL agents do not generalize to solving *any* task, even in the same environment. While related machine learning fields like computer vision and natural language processing have shown success in zero-shot (Ramesh et al., 2021) and few-shot (Radford et al., 2021) adaptation to a wide range of downstream tasks, RL lags behind in such functionalities. Unsupervised reinforcement learning aims to extract reusable information such as skills (Eysenbach et al., 2018; Zahavy et al., 2022), representations (Ghosh et al., 2023; Ma et al., 2022), world-model (Janner et al., 2022), goal-reaching policies (Agarwal et al., 2024; Sikchi et al., 2023a), etc.) from the environment using data independent of the task reward to efficiently train RL agents for any task. Recent advances in unsupervised RL (Wu et al., 2018a; Touati & Ollivier, 2021a; Blier et al., 2021; Touati et al., 2023) have shown some promise towards achieving zero-shot RL.

Recently proposed pretraining algorithms (Stooke et al., 2020; Schwarzer et al., 2021b; Sermanet et al., 2018; Nair et al., 2022; Ma et al., 2022) use self-supervised learning to learn representations from large-scale data to facilitate few-shot RL but these representations are dependent on the policies used for collecting the data. These algorithms assume that the large scale data is collected from

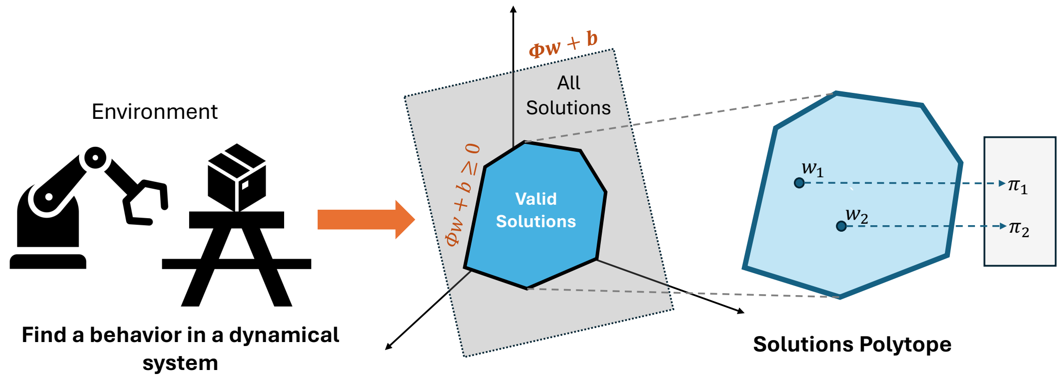


Figure 1: Method Overview: Visitation distributions corresponding to any policy must obey the Bellman Flow constraint for the dynamical system. This means they must lie on the plane defined by the the Bellman Flow equation. Being a plane, it can be represented using a set of basis set Φ and a bias. All valid (non negative) visitation distributions lie within a convex hull on this plane. The boundary of this hull is defined using the non negativity constraints: $\Phi w + b \geq 0$. Each point within this convex hull corresponds to a visitation distribution for a valid policy and is defined simply by the "coordinate" w.

a "good" policy demonstrating expert task solving behaviors. Several prior works aim to achieve generalization in multi-task RL by building upon successor features (Dayan, 1993) which represent rewards as a linear combination of state features. These methods have limited generalization capacity to unseen arbitrary tasks. Other works (Mahadevan, 2005; Bellemare et al., 2019; Farebrother et al., 2023; Machado et al., 2017a;b) represent value functions using eigenvectors of the graph Laplacian obtained from a random policy to approximate the global basis of value functions. However, the eigenvectors from a random policy cannot represent all value functions. In fact, we show that an alternative strategy of representing visitation distributions using a set of basis functions covers a larger set of solutions than doing the same with value functions. Skill learning methods (Eysenbach et al., 2018; Park et al., 2024b; Eysenbach et al., 2021) view any policy as combination of skills , but as shown by Eysenbach et al. (2021), these methods do not recover all possible skills from the MDP. Some recent works have attempted zero-shot RL by decomposing the representation of visitation distributions (Touati & Ollivier, 2021a; Touati et al., 2023), but they learn policy representations as a projection of the reward function which leads to loss of task relevant information. We present a stronger, more principled approach for representing any solution of RL in the MDP.

Any policy in the environment can be represented using visitation distributions or the distributions over states and actions that the agent visits when following a policy. We learn a basis set to represent any possible visitation distribution in the underlying environmental dynamics. We draw our inspiration from the linear programming view (Manne, 1960; Denardo, 1970; Nachum & Dai, 2020; Sikchi et al., 2023b) of reinforcement learning; the objective is to find the visitation distribution that maximizes the return (the dot-product of the visitation distribution and the reward) subject to the Bellman Flow constraints. We show that any solution of the Bellman Flow constraint for the visitation distribution can be represented as a linear combination of policy-independent basis functions and a bias. As shown in Figure 1, any visitation distribution that is a solution of the Bellman Flow for a given dynamical system lies on a plane defined using policy independent basis Φ and a bias b. On this plane, only a small convex region defines the valid (non negative) visitations distributions. Any visitation distribution in this convex hull can be obtained simply using the "coordinates" w. We introduce *Proto-Successor Measure*, the set of basis functions and bias to represent any successor measure (a generalization over visitation distributions) in the MDP that can be learnt using reward-free interaction data. At test time, obtaining the optimal policy reduces to simply finding the linear weights to combine these basis vectors that maximize its dot-product with the user-specified reward. These basis vectors only depend on the state-action transition dynamics of the MDP, independent of the initial state distribution, reward, or policy, and can be thought to compactly represent the entire dynamics.

The contributions of our work are (1) a novel, mathematically complete perspective on representation learning for Markov decision processes; (2) an efficient practical instantiation that reduces basis learning to a single-player optimization; and (3) evaluations of a number of tasks demonstrating the capability of our learned representations to quickly infer optimal policies.

2 Related Work

Unsupervised Reinforcement Learning: Unsupervised RL generally refers to a broad class of algorithms that use reward-free data to improve the efficiency of RL algorithms. We focus on methods that learn representations to produce optimal value functions for any given reward function. Representation learning through unsupervised or self-supervised RL has been discussed for both pre-training (Nair et al., 2022; Ma et al., 2022) and training as auxiliary objectives (Agarwal et al., 2021; Schwarzer et al., 2021a; Agarwal et al., 2021). While using auxiliary objectives for representation learning does accelerate policy learning for downstream tasks, the policy learning begins from scratch for a new task. Pre-training methods like Ma et al. (2022); Nair et al. (2022) use self-supervised learning techniques from computer vision like masked auto-encoding to learn representations that can be used directly for downstream tasks. These methods use large-scale datasets (Grauman et al., 2022) to learn representations but these are fitted around the policies used for collecting data. These representations do not represent any possible behavior nor are trained to represent Q functions for any reward functions. A number of works in prior literature aim to discover intents or skills using a diversity objective. These methods use the fact that the latents or skills should define the output state-visitation distributions thus diversity can be ensured by maximizing mutual information (Warde-Farley et al., 2018; Eysenbach et al., 2018; Achiam et al., 2018; Eysenbach et al., 2021) or minimizing Wasserstein distance (Park et al., 2024b) between the latents and corresponding state-visitation distributions. PSM differs from these works and takes a step towards learning representations optimal for predicting value functions as well as a zero-shot near-optimal policy for any reward.

Methods that linearize RL quantities: Learning basis vectors has been leveraged in RL to allow for transfer to new tasks. Successor features (Barreto et al., 2018) represents rewards as a linear combination of transition features and subsequently the Q-functions are linear in successor features. Several methods have extended successor features (Lehnert & Littman, 2020; Hoang et al., 2021; Alegre et al., 2022; Reinke & Alameda-Pineda, 2023) to learn better policies in more complex domains.

Spectral methods like Proto Value Functions (PVFs) (Mahadevan, 2005; Mahadevan & Maggioni, 2007) instead represent the value functions as a linear combination of basis vectors. It uses the eigenvectors of the random walk operator (graph Laplacian) as the basis vectors. Adversarial Value Functions (Bellemare et al., 2019) and Proto Value Networks (Farebrother et al., 2023) have attempted to scale up this idea in different ways. However, deriving these eigenvectors from a Laplacian is not scalable to larger state spaces. Wu et al. (2018a) recently presented an approximate scalable objective, but the Laplacian is still dependent on the policy which makes it incapable of representing all behaviors or Q functions.

Similar to our work, Forward Backward (FB) Representations (Touati & Ollivier, 2021a; Touati et al., 2023) use an inductive bias on the successor measure to decompose it into a forward and backward representation. Unlike FB, our representations are linear on a set of basis features. Additionally, FB ties the reward representation with the representation of the optimal policy derived using Q function maximization which can lead to overestimation issues and instability during training as a result of Bellman optimality backups.

3 Preliminaries

In this section we introduce some preliminaries and define terminologies that will be used in later sections. We begin with some MDP fundamentals and RL preliminaries followed by a discussion on affine spaces which form the basis for our representation learning paradigm.

3.1 Markov Decision Processes

A Markov Decision Process is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma, \mu \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \longmapsto \Delta(\mathcal{S})$ is the transition probability ($\Delta(\cdot)$ denotes a probability distribution over a set), $\gamma \in [0, 1)$ is the discount factor, μ is the distribution over initial states and $r : \mathcal{S} \times \mathcal{A} \longmapsto \mathbb{R}$ is the reward function. The *task* is specified using the reward function r and the initial state distribution μ. The goal for the RL agent is to learn a policy $\pi_\theta : \mathcal{S} \longmapsto \mathcal{A}$ that maximizes the expected return $J(\pi_\theta) = \mathbb{E}_{s_0 \sim \mu} \mathbb{E}_{\pi_\theta} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$.

In this work, we consider a *task-free* MDP which does not provide the reward function or the initial state distribution. Hence, a *task-free* or *reward-free* MDP is simply the tuple $\langle \mathcal{S}, \mathcal{A}, P, \gamma \rangle$. A *task-free* MDP essentially only captures the underlying environment dynamics and can have infinite downstream tasks specified through different reward functions.

The state-action visitation distribution, $d^\pi(s, a)$ is defined as the normalized probability of being in a state s and taking an action a if the agent follows the policy π from a state sampled from the initial state distribution. Concretely, $d^\pi(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s, a_t = a)$. A more general quantity, successor measure, $M^\pi(s, a, s^+, a^+)$, is defined as the probability of being in state s^+ and taking action a^+ when starting from the state-action pair s, a and following the policy π. Mathematically, $M^\pi(s, a, s^+, a^+) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s^+, a_t = a^+ | s_0 = s, a_0 = a)$. The state-action visitation distribution can be written as $d^\pi(s, a) = \mathbb{E}_{s_0 \sim \mu(s), a_0 \sim \pi(a_0|s_0)}[M^\pi(s_0, a_0, s, a)]$.

Both these quantities, state-action visitation distribution and successor measure, follow the Bellman Flow equations:

$$d^\pi(s, a) = (1 - \gamma)\mu(s)\pi(a|s) + \gamma \sum_{s', a' \in \mathcal{S}\mathcal{A}} P(s|s', a') d^\pi(s', a') \pi(a|s). \tag{1}$$

For successor measure, the initial state distribution changes to an identity function

$$\begin{aligned} M^\pi(s, a, s^+, a^+) = (1 - \gamma)\mathbb{1}[s = s^+, a = a^+] + \\ \gamma \sum_{s', a' \in \mathcal{S}\mathcal{A}} P(s^+|s', a') M^\pi(s, a, s', a') \pi(a^+|s^+). \end{aligned} \tag{2}$$

The RL objective has a well studied linear programming interpretation (Manne, 1960). Given any task reward function r, the RL objective can be rewritten in the form of a constrained linear program:

$$\begin{aligned} \max_d \quad & \sum_{s,a} d(s, a) r(s, a) \\ s.t. \quad & d(s, a) = (1 - \gamma)\mu(s)\pi(a|s) + \gamma \sum_{s', a' \in \mathcal{S}\mathcal{A}} P(s|s', a') d(s', a') \pi(a|s) \\ & d(s, a) \geq 0 \quad \forall s, a, \end{aligned} \tag{3}$$

and the unique policy corresponding to visitation d is obtained by $\pi(a|s) = \frac{d(s,a)}{\sum_a d(s,a)}$. The Q function can then be defined using successor measure as $Q^\pi(s, a) = \sum_{s^+, a^+} M^\pi(s, a, s+, a+) r(s^+, a^+)$ or $Q^\pi = M^\pi r$. Obtaining the optimal policies requires maximizing the Q function which requires solving $\arg\max_\pi M^\pi r$.

3.2 Affine Spaces

Let $\mathcal{V}$ be a vector space and b be a vector. An affine set is defined as $A = b + \mathcal{V} = \{x | x = b + v, v \in \mathcal{V}\}$. Any vector in a vector space can be written as a linear combination of basis vectors, i.e., $v = \sum_i^n \alpha_i v_i$ where n is the dimensionality of the vector space. This property implies that any element of an affine space can be expressed as $x = b + \sum_i^n \alpha_i v_i$. Given a system of linear equations $Ax = c$, with A being an $m \times n$ matrix ($m < n$) and $c \neq 0$, the solution x forms an affine set. Hence, there exists alphas α_i such that $x = b + \sum_i \alpha_i x_i$. The vectors $\{x_i\}$ form the basis set of the null space or *kernel* of A. The values (α_i) form the affine coordinates of x for the basis $\{x_i\}$. Hence, for a given system with known $\{x_i\}$ and b, any solution can be represented using only the affine coordinates (α_i).

4 The Basis Set for All Solutions of RL

In this section, we introduce the theoretical results that form the foundation for our representation learning approach. The goal is to learn policy-independent representations that can represent any valid visitation distribution in the environment (i.e. satisfy the Bellman Flow constraint in Equation 3). With a compact way to represent these distributions, it is possible to reduce the policy optimization problem to a search in this compact representation space. We will show that state visitation distributions and successor measures form an affine set and thus can be represented as $\sum_i \phi_i w_i^\pi + b$, where ϕ_i are basis functions, w^π are "coordinates" or weights to linearly combine the basis functions, and b is a bias term. First, we build up the formal intuition for this statement and later we will use a toy example to show how these representations can make policy search easier.

The first constraint in Equation 3 is the Bellman Flow equation. We begin with Lemma 4.1 showing that state visitation distributions that satisfy the Bellman Flow form affine sets.

Lemma 4.1. *All possible state-action visitation distributions in an MDP form an affine set.*

While Lemma 4.1 shows that any state-action visitation distribution in an MDP can be written using a linear combination of basis and bias terms, state-action visitation distributions still depend on the initial state distribution. Moreover, as shown in Equation 1, computing the state-action visitation distribution requires a summation over all states and actions in the MDP which is not always possible. Successor measures are more general than state-visitation distributions as they encode the state-action visitation of the policy conditioned on a starting state-action pair. Using similar techniques, we show that successor measures also form affine sets.

Theorem 4.2. *Any successor measure, M^π in an MDP forms an affine set and so can be represented as $\sum_i^d \phi_i w_i^\pi + b$ where ϕ_i and b are independent of the policy π and d is the dimension of the affine space.*

Following Theorem 4.2, for any w, the function $\sum_i^d \phi_i w_i^\pi + b$ will be a solution of Equation 2. Hence, given Φ (ϕ_i stacked together) and b, we do not need the first constraint on the linear program (in Equation 3) anymore. The other constraint: $\phi_i w_i + b \geq 0$ still remains which w needs to satisfy. We discuss ways to manage this constraint in Section 5.3. The linear program given a reward function now becomes,

$$\begin{aligned} \max_{w} \quad & \mathbb{E}_\mu[(\Phi w + b) r] \\ s.t. \quad & \Phi w + b \geq 0 \quad \forall s, a. \end{aligned} \tag{4}$$

In fact, any visitation distribution that is a policy-independent linear transformation of M^π, such as state visitation distribution or future state-visitation distribution, can be represented in the same way as shown in Corollary 4.3.

Corollary 4.3. *Any quantity that is a policy-independent linear transformation of M^π can be written as a linear combination of policy-independent basis and bias terms.*

Toy Example: Let's consider a simple 2 state MDP (as shown in Figure **??**) to depict how the precomputation and inference will take place. Consider the state-action visitation distribution as in Equation 1. For this simple MDP, the Φ and b can be computed using simple algebraic manipulations. For a given initial state-visitation distribution, μ and γ, the state-action visitation distribution $d = (d(s_0, a_0), d(s_1, a_0), d(s_0, a_1), d(s_1, a_1))^T$ can be written as,

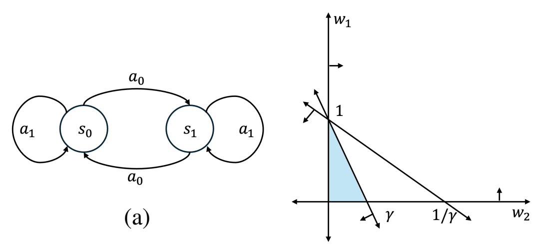
$$d = w_1 \begin{pmatrix} \frac{-\gamma}{1+\gamma} \\ \frac{-1}{1+\gamma} \\ 1 \\ 0 \end{pmatrix} + w_2 \begin{pmatrix} \frac{-1}{1+\gamma} \\ \frac{-\gamma}{1+\gamma} \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} \frac{\mu(s_0)+\gamma\mu(s_1)}{1+\gamma} \\ \frac{\mu(s_1)+\gamma\mu(s_0)}{1+\gamma} \\ 0 \\ 0 \end{pmatrix}. \tag{5}$$

The derivation for these basis vectors and the bias vector is in the supplementary material. Equation 5 represents any vector that is a solution of Equation 1 for the simple MDP. Any state-action visitation distribution possible in the MDP can now be represented using only $w = (w_1, w_2)^T$. The only constraint in the linear program of Equation 4 is $\Phi w + b \geq 0$. Looking closely, this constraint gives rise to four inequalities in w and the linear program reduces to,

$$
\begin{aligned}
\max_{w_1, w2} \quad & (\frac{-\gamma w_1 - w_2}{1+\gamma}, \frac{-w_1 - \gamma w_2}{1+\gamma}, w_1, w_2)^T r \\
s.t. \quad & w_1 + \gamma w_2 \leq \mu(s_0) + \gamma\mu(s_1) \\
& \gamma w_1 + w_2 \leq \mu(s_1) + \gamma\mu(s_0) \\
& w_1 \geq 0, w_2 \geq 0
\end{aligned}
\quad . \tag{6}
$$

The inequalities in w give rise to a simplex as shown in Figure **??**. For any specific instantiation of μ and r, the optimal policy can be easily found. For instance, if $\mu = (1,0)^T$ and the reward function, $r = (1,0,1,0)^T$, the optimal w will be obtained at the vertex $(w_1 = 1, w_2 = 0)$ and the corresponding state-action visitation distribution is $d = (0,0,1,0)^T$.

As shown for the toy MDP, the successor measures form a simplex as discussed in Eysenbach et al. (2021). Spectral Methods following Proto Value Functions (Mahadevan & Maggioni, 2007) have tried to represent value functions using a linear combination of basis vectors, $V = \Phi^{vf} w$ for some Φ^{vf}. Some prior works (Dadashi et al., 2019) have argued that value functions do not form convex polytopes. We show through Theorem 4.4 that for identical dimensionalities of basis, the span of value functions using basis functions is a subset of the set of value functions that can be represented using the span of the successor measure.



(a) (b)

Figure 2: (left) A Toy MDP with 2 states and 2 actions to depict how the linear program of RL is reduced using precomputation. (right) The corresponding simplex for w assuming the initial state distribution is $\mu = (1,0)^T$.

Theorem 4.4. *For the same dimensionality, $span\{\Phi^{vf}\}$ represents the set of the value functions spanned by Φ^{vf} and $\{span\{\Phi\}r\}$ represents the set of value functions using the successor measures spanned by Φ, $span\{\Phi^{vf}\} \subseteq \{span\{\Phi\}r\}$.*

Approaches such as Forward Backward Representations (Touati & Ollivier, 2021a) have also been based on representing successor measures but they force a latent variable z representing the policy to be a function of the reward for which the policy is optimal. The forward map that they propose is a function of this latent z. We, on the other hand, propose a representation that is truly independent of the policy or the reward.

5 METHOD

In this section, we start by introducing the practical algorithm inspired from the theory discussed in Section 4 for obtaining Φ and b. We will also discuss the inference step, i.e., obtaining w for a given reward function.

5.1 LEARNING Φ AND b

For a given policy π, its successor measure under our framework is denoted by $M^\pi = \Phi w^\pi + b$ with w^π the only object depending on policy. Given an offline dataset with density ρ, we follow prior works (Touati & Ollivier, 2021a; Blier et al., 2021) and model densities $m^\pi = M^\pi/\rho$ learned with the following objective:

$$
\begin{aligned}
L^\pi(\Phi, b, w^\pi) = & -\mathbb{E}_{s,a\sim\rho}[m^{\Phi,b,w^\pi}(s,a,s,a)] \\
& + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\rho,s^+,a^+\sim\rho}[m^{\Phi,b,w^\pi}(s,a,s^+,a^+) - \gamma\bar{m}^{\bar{\Phi},\bar{b},\bar{w}^\pi}(s',\pi(s'),s^+,a^+)]. \quad (7)
\end{aligned}
$$

The above objective only requires samples (s,a,s') from the reward-free dataset and a random state-action pair (s^+,a^+) (also sampled from the same data) to compute and minimize $L(\pi)$.

A Φ and b that allows for minimizing the $L(\pi)$ for all $\pi \in \Pi$ forms a solution to our representation learning problem. But how do we go about learning such Φ and b? A naïve way to implement learning Φ and b is via a bi-level optimization. We sample policies from the policy space of Π, for each policy we learn a w^π that optimizes the policy evaluation loss (Eq 7) and take a gradient update w.r.t Φ and b. In general, the objective can be optimized by any two-player game solving strategies with $[\Phi, b]$ as the first player and w^π as the second player. Instead, in the next section, we present an approach to simplify learning representations to a single-player game.

5.2 Simplifying Optimization via a Discrete Codebook of Policies

Learning a new w^π for each specific sampled policy π does not leverage precomputations and requires retraining from scratch. We propose parameterizing w to be conditional on policy, which allows leveraging generalization between policies that induce similar visitation and as we show, will allow us to simplify the two player game into a single player optimization. In general, policies are high-dimensional objects and compressing them can result in additional overhead. Compression by parameterizing policies with a latent variable z is another alternative but presents the challenge of covering the space of all possible policies by sampling z. Instead, we propose using a discrete codebook of policies as a way to simulate uniform sampling of all possible policies with support in the offline dataset.

Discrete Codebook of Policies: Denote z as a compact representation of policies. We propose to represent z as a random sampling *seed* that will generate a deterministic policy from the set of supported policies as follows:

$$\pi(a|s, z) = \text{Uniform Sample}(\text{seed} = z + \text{hash}(s)). \tag{8}$$

The above sampling strategy defines a unique mapping from a seed to a policy. If the seed generator is unbiased, the approach provably samples from among all possible deterministic policies uniformly. Now, with policy π_z and w_z parameterized as a function of z we derive the following single-player reduction to learn Φ, b, w jointly.

$$\texttt{PSM-objective:} \quad \arg\min_{\Phi, b, w(z)} \mathbb{E}_z[L^{\pi_z}(\Phi, b, w(z))]. \tag{9}$$

5.3 Fast Optimal Policy Inference on Downstream Tasks

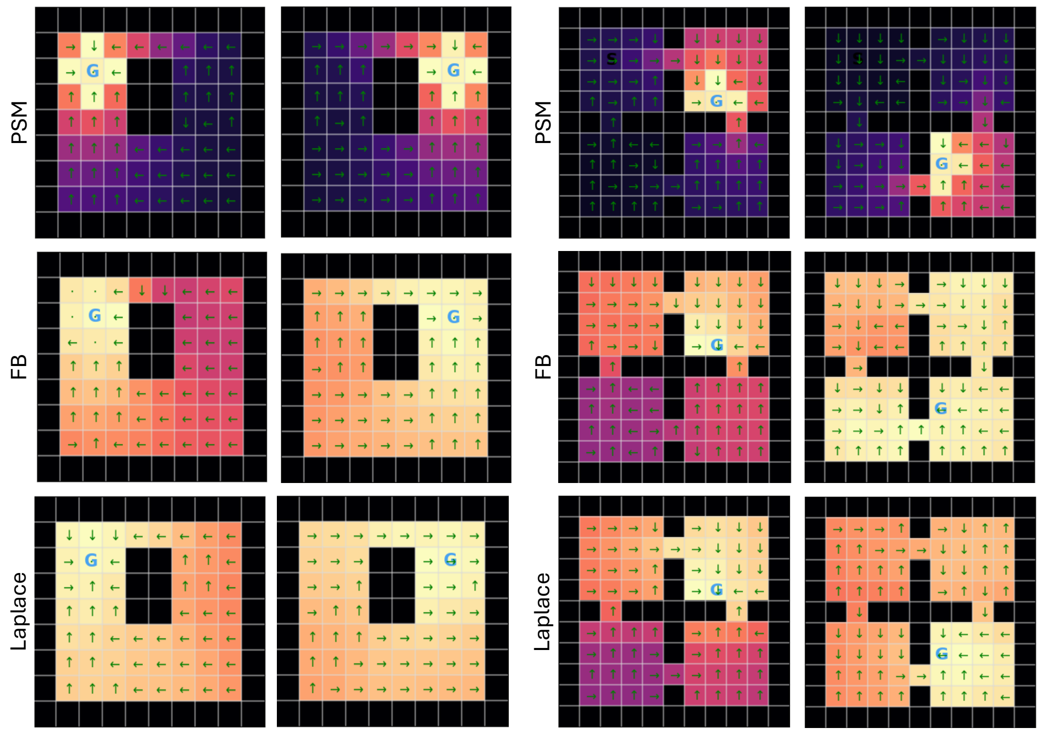
After obtaining Φ and b via the pretraining step, the only parameter to compute for obtaining the optimal Q function for a downstream task in the MDP is w. As discussed earlier, $Q^* = \max_w (\Phi w + b)r$ but simply maximizing this objective will not yield a Q function. The linear program still has a constraint of $\Phi w + b \geq 0, \forall s, a$. We solve the constrained linear program by constructing the Lagrangian dual using Lagrange multipliers $\lambda(s, a)$. The dual problem is shown in Equation 10. Here, we write the corresponding loss for the constraint as $min(\Phi w + b, 0)$.

$$\max_{\lambda \geq 0} \min_w -\Phi w r - \sum_{s,a} \lambda(s, a) min(\Phi w + b, 0). \tag{10}$$

Once w^* is obtained, the corresponding M^* and Q^* can be easily computed. The policy can be obtained as $\pi^* = \arg\max_a Q^*(s, a)$ for discrete action spaces and via DDPG style policy learning for continuous action spaces.

6 Experimental Study

Our experiments evaluate how PSM can be used to encapsulate a *task-free* MDP into a representation that will enable zero-shot inference on any downstream task. In the experiments we investigate a) the quality of value functions learned by PSM, b) the zero-shot performance of PSM in contrast to other baselines, and finally on robot manipulation task c) the ability to learn general goal-reaching skills arising from the PSM objective.



(a) Gridwold (b) Four-room

Figure 3: **Qualitative results on a gridworld and four-room:** G denotes the goal sampled for every episode. The black regions are the boundaries/obstacles. The agent needs to navigate across the grid and through the small opening (in case of four-room) to reach the goal. We visualize the optimal Q-functions inferred at test time for the given goal in the image. The arrows denote the optimal policy. (Top row) Results for PSM, (Middle Row) Results for FB, (Bottom row) Results for Laplacian Eigenfunctions.

Baselines We compare to the methods that have stood the test of time and perform best: Laplacian features (Wu et al., 2018b) and Forward-Backward (Touati et al., 2022). Laplacian features learn features of a state by considering eigenvectors of a graph Laplacian induced by a random walk. These features $\psi(s) \in \mathbb{R}^d$ obtained for each state are used to define a reward function conditioned on a reward $r(s;\psi) = \psi(s) \cdot z$ where z is sampled uniformly from a unit d-dimensional sphere. For each z an optimal policy is pretrained from the dataset on the induced reward function. During inference the corresponding z for a given reward function is obtained as a solution to the following linear regression: $\min_z \mathbb{E}_s[(\psi^\top \cdot z - r(s))^2]$. Forward-backward (FB) learns both the optimal policy and state features jointly for all reward that are in the linear span of state-features. FB methods typically assume a goal-conditioned prior during pretraining which typically helps in learning policies that reach various states in the dataset. HILP (Park et al., 2024a) makes two changes to FB: a) Reduces the tasks to be goal reaching to learn the features of a state and b) Uses a more performant offline RL method, IQL (Kostrikov et al., 2021) to learn features. We do not compare to HILP as it has been shown to have comparable performance with FB and innovates on an orthogonal axis of using a better base RL algorithm.

6.1 Zero shot Value function and Optimal Policy prediction

In this section, we consider goal-conditioned rewards on discrete gridworld and the classic four-room environments. Since the goal-conditioned rewards are state-only reward functions, we learn representations for $M^\pi(s, a, s^+)$ instead of $M^\pi(s, a, s^+, a^+)$ using the learning objectives in Equation 9.

Task Setup: Both environments have discrete state and action spaces. The action space consists of *five* actions: $\{up, right, down, left, stay\}$. We collect transitions in the environment by uniformly spawning the agent and taking a random-uniform action.This allows us toform our offline reward-free dataset will full coverage to train Φ and b. During inference, we sample a goal and infer the optimal Q function on the goal. Since the reward function is given by $r(s) = \mathbb{1}_{s=g}$, the inference looks like $Q(s,a) = \max_w \Phi(s,a,g)w \quad s.t. \quad \Phi(s,a,s')w + b(s,a,s') \geq 0 \quad \forall s,a,s'$. Figure 3 shows the Q function and the corresponding optimal policy (when executed from a fixed start state) on the gridworld and the four-room environment. As illustrated clearly, for both the environments, the optimal Q function and policy can be obtained zero-shot for any given goal-conditioned downstream task. We observe a 100% success rate on both these tasks.

Comparison to baselines: We can draw a couple of conclusions from the visualization of the Q functions inferred by the different methods. First, the Q function learnt by PSM is more sharply concentrated on optimal state-action pairs compared to the two baselines. Both baselines have more uniform value estimates, leaving only a minor differential over state values. Secondly, the baselines produce far more incorrect optimal actions (represented by the green arrows) compared to PSM.

6.2 Learning zero-shot policies for manipulation

We consider the Fetch-Reach environment with continuous states and discrete actions (Touati & Ollivier, 2021b). A dataset of size 1M is constructed using DQN+RND. FB, Laplacian and PSM all use this dataset to learn pretrained objects that can be used for zero-shot RL.

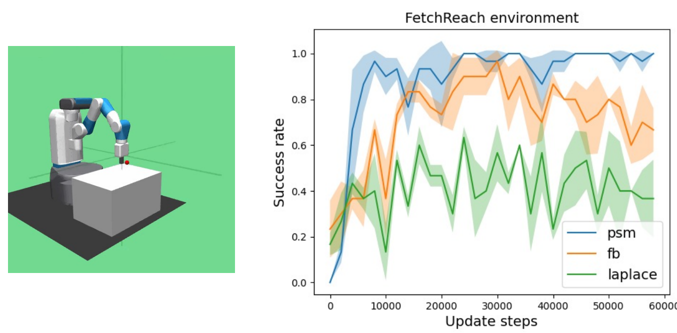


Figure 4: **Quantitative results on FetchReach:** The success rates (averaged over 3 seeds) are plotted (along with the standard deviation as shaded) with respect to the training updates for PSM, FB and Laplacian. PSM quickly reaches optimal performance while FB shows instability in maintaining its optimality. Laplacian is far from the optimal performance.

We observe that PSM outperforms baselines FB and Laplacian in its ability to learn a zero-shot policy. One key observation is that PSM learning is stable whereas FB exhibits a drop in performance, likely due to the use of Bellman optimality backups resulting in overestimation bias during training. Laplacian's capacity to output zero-shot policies is far exceeded by PSM because Laplacian methods construct the graph Laplacian for random policies and may not be able to represent optimal value functions for all rewards.

7 Conclusion

In this work, we propose Proto Successor Measures (PSM), a zero-shot RL method that compresses any MDP to allow for optimal policy inference *for any reward function* without additional environmental interactions. This framework marks a step in the direction of moving away from common idealogy in RL to solve single tasks optimally, and rather pretraining reward-free agents that are able to solve an infinite number of tasks. PSM is based on the princple that successor measures are solutions to an affine set and proposes an efficient and mathematically grounded algorithm to extract the basis for the affine set. We show that PSM can produce the optimal Q function and the optimal policy for any goal conditioned task in a number of environments outperforming prior baselines.

References

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *CoRR*, abs/1807.10299, 2018. URL http://arxiv.org/abs/1807.10299.

Siddhant Agarwal, Aaron Courville, and Rishabh Agarwal. Behavior predictive representations for generalization in reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021. URL https://openreview.net/forum?id=b5PJaxS6Jxg.

Siddhant Agarwal, Ishan Durugkar, Peter Stone, and Amy Zhang. f-policy gradients: A general framework for goal-conditioned rl using f-divergences. *Advances in Neural Information Processing Systems*, 36, 2024.

Lucas Nunes Alegre, Ana Bazzan, and Bruno C Da Silva. Optimistic linear support and successor features as a basis for optimal policy transfer. In *International conference on machine learning*, pp. 394–413. PMLR, 2022.

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning, 2018.

Marc Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *CoRR*, abs/2101.07123, 2021. URL https://arxiv.org/abs/2101.07123.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Robert Dadashi, Adrien Ali Taïga, Nicolas Le Roux, Dale Schuurmans, and Marc G. Bellemare. The value function polytope in reinforcement learning. *CoRR*, abs/1901.11524, 2019. URL http://arxiv.org/abs/1901.11524.

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

Eric V Denardo. On linear programming in a markov decision problem. *Management Science*, 16(5): 281–288, 1970.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *CoRR*, abs/1802.06070, 2018. URL http://arxiv.org/abs/1802.06070.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The information geometry of unsupervised reinforcement learning. *arXiv preprint arXiv:2110.02719*, 2021.

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. *arXiv preprint arXiv:2304.12567*, 2023.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Dibya Ghosh, Chethan Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions, 2023. URL https://arxiv.org/abs/2304.04782.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18995–19012, 2022.

Christopher Hoang, Sungryull Sohn, Jongwook Choi, Wilka Carvalho, and Honglak Lee. Successor feature landmarks for long-horizon goal-conditioned reinforcement learning, 2021.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Lucas Lehnert and Michael L. Littman. Successor features combine elements of model-free and model-based reinforcement learning. *Journal of Machine Learning Research*, 21(196):1–53, 2020. URL http://jmlr.org/papers/v21/19-060.html.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Marlos C. Machado, Marc G. Bellemare, and Michael H. Bowling. A laplacian framework for option discovery in reinforcement learning. *CoRR*, abs/1703.00956, 2017a. URL http://arxiv.org/abs/1703.00956.

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *CoRR*, abs/1710.11089, 2017b. URL http://arxiv.org/abs/1710.11089.

Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 553–560, 2005.

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), 2007.

Alan S Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Ofir Nachum and Bo Dai. Reinforcement learning via fenchel-rockafellar duality. *arXiv preprint arXiv:2001.01866*, 2020.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation, 2022.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations, 2024a.

Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction, 2024b.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021. URL https://arxiv.org/abs/2102.12092.

Chris Reinke and Xavier Alameda-Pineda. Successor feature representations, 2023.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations, 2021a.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R. Devon Hjelm, Philip Bachman, and Aaron C. Courville. Pretraining representations for data-efficient reinforcement learning. *CoRR*, abs/2106.04799, 2021b. URL `https://arxiv.org/abs/2106.04799`.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video, 2018.

Harshit Sikchi, Rohan Chitnis, Ahmed Touati, Alborz Geramifard, Amy Zhang, and Scott Niekum. Score models for offline goal-conditioned reinforcement learning. *arXiv preprint arXiv:2311.02013*, 2023a.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. *arXiv preprint arXiv:2302.08560*, 2023b.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *CoRR*, abs/2009.08319, 2020. URL `https://arxiv.org/abs/2009.08319`.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021a.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021b.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? *arXiv preprint arXiv:2209.14935*, 2022.

Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist?, 2023.

David Warde-Farley, Tom Van de Wiele, Tejas D. Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *CoRR*, abs/1811.11359, 2018. URL `http://arxiv.org/abs/1811.11359`.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in RL: learning representations with efficient approximations. *CoRR*, abs/1810.04586, 2018a. URL `http://arxiv.org/abs/1810.04586`.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018b.

Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896): 223–228, 2022.

Tom Zahavy, Yannick Schroecker, Feryal Behbahani, Kate Baumli, Sebastian Flennerhag, Shaobo Hou, and Satinder Singh. Discovering policies with domino: Diversity optimization maintaining near optimality. *arXiv preprint arXiv:2205.13521*, 2022.

APPENDIX

A THEORETICAL RESULTS

In this section, we will present the proofs for all the Lemmas and Theorems stated in Section 4.

A.1 PROOF OF LEMMA 4.1

Lemma 4.1. All possible state-action visitation distributions in an MDP form an affine set.

Proof. Any state-action visitation distribution, $d^(s,a)$ must satsify the Bellman Flow equation:

$$\sum_a d^\pi(s,a) = (1-\gamma)\mu(s) + \gamma \sum_{s',a'} \mathbb{P}(s|s',a') d^\pi(s',a'). \tag{11}$$

This equation can be written in matrix notation as:

$$\sum_a d^\pi = (1-\gamma)\mu + \gamma P^T d^\pi. \tag{12}$$

Rearranging the terms,

$$(S - \gamma P^T) d^\pi = (1-\gamma)\mu, \tag{13}$$

where S is the matrix for $\sum_a$. This equation is an affine equation of the form $Ax = b$ whose solution set forms an affine set. Hence all state-visitation distributions d^π form an affine set.

□

A.2 PROOF OF THEOREM 4.2

Theorem 4.2. Any successor measure, M^π, in an MDP forms an affine set and so can be represented as $\sum_i^d \phi_i w_i^\pi + b$ where ϕ_i and b are independent of the policy π and d is the dimension of the affine space.

Proof. Using Lemma 4.1, we have shown that state-action visitation distributions form affine sets. Similarly, successor measures, $M^\pi(s,a,s^+,a^+)$ are solutions of the Bellman Flow equation:

$$M^\pi(s,a,s^+,a^+) = (1-\gamma)\mathbb{1}[s=s^+, a=a^+] + \gamma \sum_{s',a' \in \mathcal{SA}} P(s^+|s',a') M^\pi(s,a,s',a') \pi(a^+|s^+). \tag{14}$$

Taking summation over a^+ on both sides gives us an equation very similar to Equation 11 and so can be written by rearranging as,

$$(S - \gamma P^T) M^\pi = (1-\gamma)\mathbb{1}[s=s^+]. \tag{15}$$

With similar arguments as in Lemma 4.1, M^π also forms an affine set. Any element x of an affine set can be written as $\sum_i^d \phi_i w_i + b$ where $\langle \phi_i \rangle$ are the basis and b is a bias vector. The basis is given by the null space of the matrix operator $(S - \gamma P^T)$. Since the operator $(S - \gamma P^T)$ and the vector $(1-\gamma)\mathbb{1}[s=s^+]$ are independent of the policy, the basis Φ and the bias b are also independent of the policy. □

A.3 PROOF OF THEOREM 4.4

Theorem 4.4. For the same dimensionality, $span\{\Phi^{vf}\}$ represents the set of the value functions spanned by Φ^{vf} and $\{span\{\Phi\}r\}$ represents the set of value functions using the successor measures spanned by Φ, $span\{\Phi^{vf}\} \subseteq \{span\{\Phi\}r\}$.

Proof. We need to show that any element that belongs to the set $\{span\{\Phi\}r\}$ also belongs to the set $span\{\Phi^{vf}\}$.

$$V^\pi(s) = \sum_i \beta_i^\pi \Phi_i^{vf}(s).$$

If we assume a special $\Phi_i(s, s') = \sigma_i(s)\eta_i(s')$,

$$\begin{aligned} V^\pi(s) &= \sum_i w_i^\pi \sum_{s'} \Phi(s, s')r(s') \\ &= \sum_i \left[w_i^\pi \sum_{s'} \eta_i(s')r(s')\right]\sigma_i(s). \end{aligned}$$

The two equations match with $\beta_i^\pi = w_i^\pi \sum_{s'} \eta_i(s')r(s')$ and $\sigma_i(s) = \Phi_i^{vf}(s)$. This implies for every instance in the span of Φ^{vf}, there exists some instance in the span of Φ. $\square$

B EXPERIMENTAL DETAILS

B.1 GRIDWORLDS

We use `https://github.com/facebookresearch/controllable_agent` code-base to build upon the gridworld and 4 room experiments. The baseline FB is already implemented in the repository. As discused in Touati et al. (2023), implementing the Laplacian baselines require a few lines of modification to the FB code. We implement the Laplacian method accordingly in the code-base. The exploratory data is collected by uniformly spawning the agent and taking a random action. Each of the three method is trained on the reward-free exploratory data. At test time, a random goal is sampled and the optimal Q function is inferred by each. The plots in Figure 3 show the optimal value $V^*(s) = \max_a Q^*(s, a)$ for every state and the optimal action $a^* = \arg\max_a Q^*(s, a)$ is marked using a green arrow.

The state representation is given by (x, y) which are scaled down to be in $[0, 1]$. The action space consists of *five* actions: $\{up, right, down, left, stay\}$.

B.2 FETCH

We build on top of `https://github.com/ahmed-touati/controllable_agent` which contains the Fetch environments with discretized action spaces. The state space is unchanged but the action space is discretized to produce manhattan style movements i.e. move one-coordinate at a time. These six actions are mapped to the true actions of Fetch as: $\{0 : [1, 0, 0, 0], 1 : [0, 1, 0, 0], 2 : [0, 0, 1, 0], 3 : [-1, 0, 0, 0], 4 : [0, -1, 0, 0], 5 : [0, 0, -1, 0]\}$. The exploratory data is collected by running DQN (Mnih et al., 2013) training with RND reward (Burda et al., 2018) taken from `https://github.com/iDurugkar/adversarial-intrinsic-motivation`. 20000 trajectories, each of length 50, are collected.

For the quantitative analysis, the dimensionality of the basis (in case of PSM) or the embedding space (in case of FB, Laplacian) is set to 100. All the methods use the learning rate is 0.0001 and $\gamma = 0.99$.