A Geometric Perspective on Optimal Representations for Reinforcement Learning

Marc G. Bellemare[1], **Will Dabney**[2], **Robert Dadashi**[1], **Adrien Ali Taiga**[1,3], **Pablo Samuel Castro**[1], **Nicolas Le Roux**[1], **Dale Schuurmans**[1,4], **Tor Lattimore**[2], **Clare Lyle**[5]

Abstract

We propose a new perspective on representation learning in reinforcement learning based on geometric properties of the space of value functions. We leverage this perspective to provide formal evidence regarding the usefulness of value functions as auxiliary tasks. Our formulation considers adapting the representation to minimize the (linear) approximation of the value function of all stationary policies for a given environment. We show that this optimization reduces to making accurate predictions regarding a special class of value functions which we call *adversarial value functions* (AVFs). We demonstrate that using value functions as auxiliary tasks corresponds to an expected-error relaxation of our formulation, with AVFs a natural candidate, and identify a close relationship with proto-value functions (Mahadevan, 2005). We highlight characteristics of AVFs and their usefulness as auxiliary tasks in a series of experiments on the four-room domain.

1 Introduction

A good representation of state is key to practical success in reinforcement learning. While early applications used hand-engineered features (Samuel, 1959), these have proven onerous to generate and difficult to scale. As a result, methods in representation learning have flourished, ranging from basis adaptation (Menache et al., 2005; Keller et al., 2006), gradient-based learning (Yu and Bertsekas, 2009), proto-value functions (Mahadevan and Maggioni, 2007), feature generation schemes such as tile coding (Sutton, 1996) and the domain-independent features used in some Atari 2600 game-playing agents (Bellemare et al., 2013; Liang et al., 2016), and nonparametric methods (Ernst et al., 2005; Farahmand et al., 2016; Tosatto et al., 2017). Today, the method of choice is deep learning. Deep learning has made its mark by showing it can learn complex representations of relatively unprocessed inputs using gradient-based optimization (Tesauro, 1995; Mnih et al., 2015; Silver et al., 2016).

Most current deep reinforcement learning methods augment their main objective with additional losses called *auxiliary tasks*, typically with the aim of facilitating and regularizing the representation learning process. The UNREAL algorithm, for example, makes predictions about future pixel values (Jaderberg et al., 2017); recent work approximates a one-step transition model to achieve a similar effect (François-Lavet et al., 2018; Gelada et al., 2019). The good empirical performance of distributional reinforcement learning (Bellemare et al., 2017) has also been attributed to representation learning effects, with recent visualizations supporting this claim (Such et al., 2019). However, while there is now conclusive empirical evidence of the usefulness of auxiliary tasks, their design and justification remain on the whole ad-hoc. One of our main contributions is to provides a formal framework in which to reason about auxiliary tasks in reinforcement learning.

We begin by formulating an optimization problem whose solution is a form of optimal representation. Specifically, we seek a state representation from which we can best approximate the value function of any stationary policy for a given Markov Decision Process. Simultaneously, the largest approximation

[1]Google Brain [2]DeepMind [3]Mila, Université de Montréal [4]University of Alberta [5]University of Oxford

Preprint. Under review.

error in that class serves as a measure of the quality of the representation. While our approach may appear naive – in real settings, most policies are uninteresting and hence may distract the representation learning process – we show that our representation learning problem can in fact be restricted to a special subset of value functions which we call *adversarial value functions* (AVFs). We then characterize these adversarial value functions and show they correspond to deterministic policies that either minimize or maximize the expected return at each state, based on the solution of a network-flow optimization derived from an interest function δ.

A consequence of our work is to formalize why predicting value function-like objects is helpful in learning representations, as has been argued in the past (Sutton et al., 2011, 2016). We show how using these predictions as auxiliary tasks can be interpreted as a relaxation of our optimization problem. From our analysis, we hypothesize that auxiliary tasks that resemble adversarial value functions should give rise to good representations in practice. We complement our theoretical results with an empirical study in a simple grid world environment, focusing on the use of deep learning techniques to learn representations. We find that predicting adversarial value functions as auxiliary tasks leads to rich representations.

2 Setting

We consider an environment described by a Markov Decision Process $\langle \mathcal{X}, \mathcal{A}, r, P, \gamma \rangle$ (Puterman, 1994); $\mathcal{X}$ and $\mathcal{A}$ are finite state and action spaces, $P : \mathcal{X} \times \mathcal{A} \to \mathscr{P}(\mathcal{X})$ is the transition function, γ the discount factor, and $r : \mathcal{X} \to \mathbb{R}$ the reward function. For a finite set $\mathcal{S}$, write $\mathscr{P}(\mathcal{S})$ for the probability simplex over $\mathcal{S}$. A (stationary) *policy* π is a mapping $\mathcal{X} \to \mathscr{P}(\mathcal{A})$, also denoted $\pi(a \,|\, x)$. We denote the set of policies by $\mathcal{P} = \mathscr{P}(\mathcal{A})^{\mathcal{X}}$. We combine a policy π with the transition function P to obtain the state-to-state transition function $P^\pi(x' \,|\, x) := \sum_{a \in \mathcal{A}} \pi(a \,|\, x) P(x' \,|\, x, a)$. The *value function* V^π describes the expected discounted sum of rewards obtained by following π:

$$V^\pi(x) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(x_t) \,\Big|\, x_0 = x, x_{t+1} \sim P^\pi(\cdot \,|\, x_t)\Big].$$

The value function satisfies Bellman's equation (Bellman, 1957): $V^\pi(x) = r(x) + \gamma \, \mathbb{E}_{P^\pi} V^\pi(x')$. We will find it convenient to use vector notation: Assuming there are $n = |\mathcal{X}|$ states, we view r and V^π as vectors in $\mathbb{R}^n$ and $P^\pi \in \mathbb{R}^{n \times n}$, yielding

$$V^\pi = r + \gamma P^\pi V^\pi = (I - \gamma P^\pi)^{-1} r.$$

A *d-dimensional representation* is a mapping $\phi : \mathcal{X} \to \mathbb{R}^d$; $\phi(x)$ is the *feature vector* for state x. We write $\Phi \in \mathbb{R}^{n \times d}$ to denote the matrix whose rows are $\phi(\mathcal{X})$, and with some abuse of notation denote the set of d-dimensional representations by $\mathscr{R} \equiv \mathbb{R}^{n \times d}$. For a given representation and weight vector $\theta \in \mathbb{R}^d$, the linear approximation for a value function is

$$\hat{V}_{\phi,\theta}(x) := \phi(x)^\top \theta.$$

We consider the approximation minimizing the uniformly weighted squared error

$$\big\|\hat{V}_{\phi,\theta} - V^\pi\big\|_2^2 = \sum_{x \in \mathcal{X}} (\phi(x)^\top \theta - V^\pi(x))^2.$$

We denote by $\hat{V}^\pi_\phi$ the projection of V^π onto the linear subspace $H = \big\{\Phi\theta : \theta \in \mathbb{R}^d\big\}$.

2.1 Two-Part Approximation

We view $\hat{V}^\pi_\phi$ as a *two-part approximation* arising from the composition of an adjustable representation ϕ and a weight vector θ; we use the term "two-part" to emphasize that the mapping $\phi(x) \mapsto \hat{V}^\pi_\phi(x)$ is linear, while ϕ itself may not be. This separation into two parts gives us a simple framework in which to study the behaviour of representation learning, in particular deep networks applied to reinforcement learning. We will further consider the use of $\phi(x)$ to make additional predictions, called *auxiliary tasks* following common usage, and whose purpose is to improve or stabilize the representation.

We study two-part approximations in an idealized setting where the length d of $\phi(x)$ is fixed and smaller than n, but the mapping is otherwise unconstrained. Even this idealized design offers

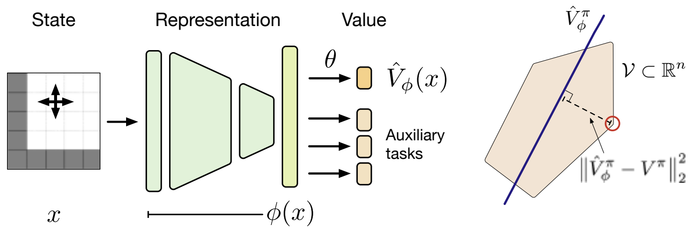

Figure 1: **Left.** A deep reinforcement learning architecture viewed as a two-part approximation. **Right.** The optimal representation ϕ^* is a linear subspace that cuts through the value polytope.

interesting problems to study. We might be interested in sharing a representation across problems, as is often done in transfer or continual learning. In this context, auxiliary tasks may inform how the value function should generalize to these new problems. In many problems of interest, the weights θ can also be optimized more efficiently than the representation itself, warranting the view that the representation should be adapted using a different process (Levine et al., 2017; Chung et al., 2019).

Note that a trivial "value-as-feature" representation exists for the single-policy optimization problem

$$\min_{\phi \in \mathscr{R}} \left\| \hat{V}^\pi_\phi - V^\pi \right\|_2^2;$$

this approximation sets $\phi(x) = V^\pi(x), \theta = 1$. In this paper we take the stance that this is not a satisfying representation, and that a good representation should be in the service of a broader goal (e.g. control, transfer, or fairness).

3 Representation Learning by Approximating Value Functions

We measure the quality of a representation ϕ in terms of how well it can approximate all possible value functions, formalized as the *representation error*

$$L(\phi) := \max_{\pi \in \mathcal{P}} L(\phi; \pi), \quad L(\phi; \pi) := \left\| \hat{V}^\pi_\phi - V^\pi \right\|_2^2.$$

We consider the problem of finding the representation $\phi \in \mathscr{R}$ minimizing $L(\phi)$:

$$\min_{\phi \in \mathscr{R}} \max_{\pi \in \mathcal{P}} \left\| \hat{V}^\pi_\phi - V^\pi \right\|_2^2. \tag{1}$$

In the context of our work, we call this the *representation learning problem* (RLP) and say that a representation ϕ^* is *optimal* when it minimizes the error in (1). Note that $L(\phi)$ (and hence ϕ^*) depends on characteristics of the environment, in particular on both reward and transition functions.

We consider the RLP from a geometric perspective (Figure 1, right). Dadashi et al. (2019) showed that the set of value functions achieved by the set of policies $\mathcal{P}$, denoted

$$\mathcal{V} := \{V \in \mathbb{R}^n : V = V^\pi \text{ for some } \pi \in \mathcal{P}\},$$

forms a (possibly nonconvex) polytope. As previously noted, a given representation ϕ defines a linear subspace H of possible value approximations. The maximal error is achieved by the value function in $\mathcal{V}$ which is furthest along the subspace normal to H, since $\hat{V}^\pi_\phi$ is the orthogonal projection of V^π.

We say that $V \in \mathcal{V}$ is an *extremal vertex* if it is a vertex of the convex hull of $\mathcal{V}$. Our first result shows that for any direction $\delta \in \mathbb{R}^n$, the furthest point in $\mathcal{V}$ along δ is an extremal vertex, and is in general unique for this δ (proof in the appendix).

Lemma 1. *Let $\delta \in \mathbb{R}^n$ and define the functional $f_\delta(V) := \delta^\top V$, with domain $\mathcal{V}$. Then f_δ is maximized by an extremal vertex $U \in \mathcal{V}$, and there is a deterministic policy π for which $V^\pi = U$. Furthermore, the set of directions $\delta \in \mathbb{R}^n$ for which the maximum of f_δ is achieved by multiple extremal vertices has Lebesgue measure zero in $\mathbb{R}^n$.*

Denote by $\mathcal{P}_v$ the set of policies corresponding to extremal vertices of $\mathcal{V}$. We next derive an equivalence between the RLP and an optimization problem which only considers policies in $\mathcal{P}_v$.

Theorem 1. *For any representation $\phi \in \mathscr{R}$, the maximal approximation error measured over all value functions is the same as the error measured over the set of extremal vertices:*

$$\max_{\pi \in \mathcal{P}} \left\| \hat{V}_\phi^\pi - V^\pi \right\|_2^2 = \max_{\pi \in \mathcal{P}_v} \left\| \hat{V}_\phi^\pi - V^\pi \right\|_2^2.$$

Theorem 1 indicates that we can find an optimal representation by considering a finite (albeit exponential) number of value functions, since each extremal vertex corresponds to the value function of some deterministic policy, of which there are at most an exponential number. We will call these *adversarial value functions* (AVFs), because of the minimax flavour of the RLP.

Solving the RLP allows us to provide quantifiable guarantees on the performance of certain value-based learning algorithms. For example, in the context of least-squares policy iteration (LSPI; Lagoudakis and Parr, 2003), minimizing the representation error L directly improves the performance bound. By contrast, we cannot have the same guarantee if ϕ is learned by minimizing the approximation error for a single value function.

Corollary 1. *Let ϕ^* be an optimal representation in the* RLP. *Consider the sequence of policies $\pi_0, \pi_1, \dots$ derived from LSPI using ϕ^* to approximate $V^{\pi_0}, V^{\pi_1}, \dots$ under a uniform sampling of the state-space. Then there exists an MDP-dependent constant $C \in \mathbb{R}$ such that*

$$\limsup_{k \to \infty} \left\| V^* - V^{\pi_k} \right\|_2^2 \leq C L(\phi^*).$$

This result is a direct application of the quadratic norm bounds given by Munos (2003), in whose work the constant is made explicit. We emphasize that the result is illustrative; our approach should enable similar guarantees in other contexts (e.g. Munos, 2007; Petrik and Zilberstein, 2011).

3.1 The Structure of Adversarial Value Functions

The RLP suggests that an agent trained to predict various value functions should develop a good state representation. Intuitively, one may worry that there are simply too many "uninteresting" policies, and that a representation learned from their value functions emphasizes the wrong quantities. However, the search for an optimal representation ϕ^* is closely tied to the much smaller set of adversarial value functions (AVFs). The aim of this section is to characterize the structure of AVFs and show that they form an *interesting* subset of all value functions. From this, we argue that their use as auxiliary tasks should also produce structured representations.

From Lemma 1, recall that an AVF is geometrically defined using a vector $\delta \in \mathbb{R}^n$ and the functional $f_\delta(V) := \delta^\top V$, which the AVF maximizes. Since f_δ is restricted to the value polytope, we can consider the equivalent policy-space functional $g_\delta : \pi \mapsto \delta^\top V^\pi$. Observe that

$$\max_{\pi \in \mathcal{P}} g_\delta(\pi) = \max_{\pi \in \mathcal{P}} \delta^\top V^\pi = \max_{\pi \in \mathcal{P}} \sum_{x \in \mathcal{X}} \delta(x) V^\pi(x). \tag{2}$$

In this optimization problem, the vector δ defines a weighting over the state space $\mathcal{X}$; for this reason, we call δ an *interest function* in the context of AVFs. Whenever $\delta \geq 0$ componentwise, we recover the optimal value function, irrespective of the exact magnitude of δ (Bertsekas, 2012). If $\delta(x) < 0$ for some x, however, the maximization becomes a minimization. As the next result shows, the policy maximizing $f_\delta(\pi)$ depends on a network flow d_π derived from δ and the transition function P.

Theorem 2. *Maximizing the functional g_δ is equivalent to finding a network flow d_π that satisfies a reverse Bellman equation:*

$$\max_{\pi \in \mathcal{P}} \delta^\top V^\pi = \max_{\pi \in \mathcal{P}} d_\pi^\top r, \qquad d_\pi = \delta + \gamma P^{\pi\top} d_\pi.$$

For a policy $\tilde{\pi}$ maximizing the above we have

$$V^{\tilde{\pi}}(x) = r(x) + \gamma \begin{cases} \max_{a \in \mathcal{A}} \mathbb{E}_{x' \sim P} V^{\tilde{\pi}}(x') & d_{\tilde{\pi}}(x) > 0, \\ \min_{a \in \mathcal{A}} \mathbb{E}_{x' \sim P} V^{\tilde{\pi}}(x') & d_{\tilde{\pi}}(x) < 0. \end{cases}$$

Corollary 2. *There are at most 2^n distinct adversarial value functions.*

The vector d_π corresponds to the sum of discounted interest weights flowing through a state x, similar to the dual variables in the theory of linear programming for MDPs (Puterman, 1994). Theorem 2, by way of the corollary, implies that there are fewer AVFs ($\leq 2^n$) than deterministic policies ($= |\mathcal{A}|^n$). It also implies that AVFs relate to a reward-driven purpose, similar to how the optimal value function describes the goal of maximizing return. We will illustrate this point empirically in Section 4.1.

3.2 Relationship to Auxiliary Tasks

So far we have argued that solving the RLP leads to a representation which is optimal in a meaningful sense. However, solving the RLP seems computationally intractable: there are an exponential number of deterministic policies to consider (Prop. 1 in the appendix gives a quadratic formulation with quadratic constraints). Using interest functions does not mitigate this difficulty: the computational problem of finding the AVF for a single interest function is NP-hard, even when restricted to deterministic MDPs (Prop. 2 in the appendix).

Instead, in this section we consider a relaxation of the RLP and show that this relaxation describes existing representation learning methods, in particular those that use auxiliary tasks. Let ξ be some distribution over $\mathbb{R}^n$. We begin by replacing the maximum in (1) by an expectation:

$$\min_{\phi \in \mathscr{R}} \mathop{\mathbb{E}}_{V \sim \xi} \big\| \hat{V}_\phi - V \big\|_2^2 . \tag{3}$$

The use of the expectation offers three practical advantages over the use of the maximum. First, this leads to a differentiable objective which can be minimized using deep learning techniques. Second, the choice of ξ gives us an additional degree of freedom; in particular, ξ needs not be restricted to the value polytope. Third, the minimizer in (3) is easily characterized, as the following theorem shows.

Theorem 3. *Let $u_1^*, \dots, u_d^* \in \mathbb{R}^n$ be the principal components of the distribution ξ, in the sense that*

$$u_i^* := \mathop{\arg\max}_{u \in B_i} \mathop{\mathbb{E}}_{V \sim \xi} (u^\top V)^2, \text{ where } B_i := \{u \in \mathbb{R}^n : \|u\|_2^2 = 1, u^\top u_j^* = 0 \ \forall j < i\}.$$

Equivalently, $u_1^*, \dots, u_d^*$ are the eigenvectors of $\mathbb{E}_\xi VV^\top \in \mathbb{R}^{n \times n}$ with the d largest eigenvalues. Then the matrix $[u_1^*, \dots, u_d^*] \in \mathbb{R}^{n \times d}$, viewed as a map $\mathcal{X} \to \mathbb{R}^d$, is a solution to (3). When the principal components are uniquely defined, any minimizer of (3) spans the same subspace as $u_1^*, \dots, u_d^*$.

One may expect the quality of the learned representation to depend on how closely the distribution ξ relates to the RLP. From an auxiliary tasks perspective, this corresponds to choosing tasks that are in some sense useful. For example, generating value functions from the uniform distribution over the set of policies $\mathcal{P}$, while a natural choice, may put too much weight on "uninteresting" value functions.

In practice, we may further restrict ξ to a finite set $\boldsymbol{V}$. Under a uniform weighting, this leads to a *representation loss*

$$L(\phi; \boldsymbol{V}) := \sum_{V \in \boldsymbol{V}} \big\| \hat{V}_\phi - V \big\|_2^2 \tag{4}$$

which corresponds to the typical formulation of an auxiliary-task loss (e.g. Jaderberg et al., 2017). In a deep reinforcement learning setting, one typically minimizes (4) using stochastic gradient descent methods, which scale better than batch methods such as singular value decomposition (but see Wu et al. (2019) for further discussion).

Our analysis leads us to conclude that, in many cases of interest, the use of auxiliary tasks produces representations that are close to the principal components of the set of tasks under consideration. If $\boldsymbol{V}$ is well-aligned with the RLP, minimizing $L(\phi; \boldsymbol{V})$ should give rise to a reasonable representation. To demonstrate the power of this approach, in Section 4 we will study the case when the set $\boldsymbol{V}$ is constructed by sampling AVFs – emphasizing the policies that support the solution to the RLP.

3.3 Relationship to Proto-Value Functions

Proto-value functions (Mahadevan and Maggioni, 2007, PVF) are a family of representations which vary smoothly across the state space. Although the original formulation defines this representation as the largest-eigenvalue eigenvectors of the Laplacian of the transition function's graphical structure, recent formulations use the top singular vectors of $(I - \gamma P^\pi)^{-1}$, where π is the uniformly random policy (Stachenfeld et al., 2014; Machado et al., 2017; Behzadian and Petrik, 2018).

In line with the analysis of the previous section, proto-value functions can also be interpreted as defining a set of value-based auxiliary tasks. Specifically, if we define an indicator reward function $r_y(x) := \mathbb{I}_{[x=y]}$ and a set of value functions $\boldsymbol{V} = \{(I - \gamma P^\pi)^{-1} r_y\}_{y \in \mathcal{X}}$ with π the uniformly random policy, then any d-dimensional representation that minimizes (4) spans the same basis as the d-dimensional PVF (up to the bias term). This suggests a connection with hindsight experience replay (Andrychowicz et al., 2017), whose auxiliary tasks consists in reaching previously experienced states.

4 Empirical Studies

In this section we complement our theoretical analysis with an experimental study. In turn, we take a closer look at 1) the **structure** of adversarial value functions, 2) the **shape of representations** learned using AVFs, and 3) the **performance profile** of these representations in a control setting.

Our eventual goal is to demonstrate that the representation learning problem (1), which is based on approximating value functions, gives rise to representations that are both interesting and comparable to previously proposed schemes. Our concrete instantiation (Algorithm 1) uses the representation loss (4). As-is, this algorithm is of limited practical relevance (our AVFs are learned using a tabular representation) but we believe provides an inspirational basis for further developments.

Algorithm 1 Representation learning using AVFs

```
input  $k$ – desired number of AVFs, $d$ – desired number of features.
  Sample $\delta_1, \ldots, \delta_k \sim [-1, 1]^n$
  Compute $\mu_i = \arg\max_\pi \delta_i^\top V^\pi$ using a policy gradient method
  Find $\phi^* = \arg\min_\phi L(\phi; \{V^{\mu_1}, \ldots, V^{\mu_k}\})$ (Equation 4)
```

We perform all of our experiments within the four-room domain (Sutton et al., 1999; Solway et al., 2014; Machado et al., 2017, Figure 2, see also Appendix H.1).

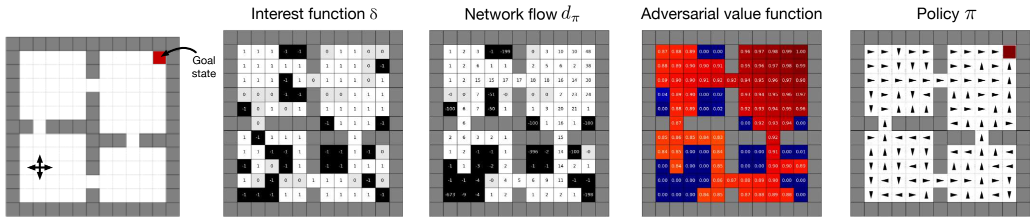

Figure 2: **Leftmost.** The four-room domain. **Other panels.** An interest function δ, the network flow d_π, the corresponding adversarial value function (blue/red = low/high value) and its policy.

We consider a two-part approximation where we pretrain ϕ end-to-end to predict a set of value functions. Our aim here is to compare the effects of using different sets of value functions, including AVFs, on the learned representation. As our focus is on the efficient use of a d-dimensional representation (with $d < n$, the number of states), we encode individual states as one-hot vectors and map them into $\phi(x)$ without capacity constraints. Additional details may be found in Appendix H.

4.1 Adversarial Value Functions

Our first set of results studies the structure of adversarial value functions in the four-room domain. We generated interest functions by assigning a value $\delta(x) \in \{-1, 0, 1\}$ uniformly at random to each state x (Figure 2, left). We restricted δ to these discrete choices for illustrative purposes.

We then used model-based policy gradient (Sutton et al., 2000) to find the policy maximizing $\sum_{x \in \mathcal{X}} \delta(x) V^\pi(x)$. We observed some local minima or accumulation points but as a whole reasonable solutions were found. The resulting network flow and AVF for a particular sample are shown in Figure 2. For most states, the signs of δ and d_π agree; however, this is not true of all states (larger version and more examples in appendix, Figures 6, 7). As expected, states for which $d_\pi > 0$ (respectively, $d_\pi < 0$) correspond to states maximizing (resp. minimizing) the value function. Finally, we remark on the "flow" nature of d_π: trajectories over minimizing states accumulate in corners or loops, while those over maximizing states flow to the goal. We conclude that AVFs exhibit interesting structure, and are generated by policies that are not random (Figure 2, right). As we will see next, this is a key differentiator in making AVFs good auxiliary tasks.

4.2 Representation Learning with AVFs

We next consider the representations that arise from training a deep network to predict AVFs (denoted AVF from here on). We sample $k = 1000$ interest functions and use Algorithm 1 to generate k AVFs.

We combine these AVFs into the representation loss (4) and adapt the parameters of the deep network using Rmsprop (Tieleman and Hinton, 2012).

We contrast the AVF-driven representation with one learned by predicting the value function of random deterministic policies (RP). Specifically, these policies are generated by assigning an action uniformly at random to each state. We also consider the value function of the uniformly random policy (VALUE). While we make these choices here for concreteness, other experiments yielded similar results (e.g. predicting the value of the optimal policy; appendix, Figure 8). In all cases, we learn a $d = 16$ dimensional representation, not including the bias unit.

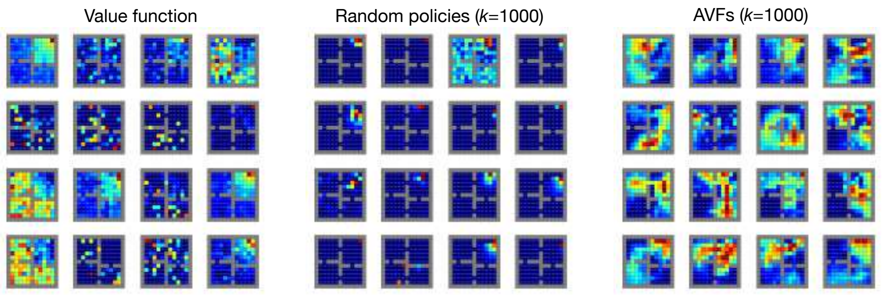

Figure 3: 16-dimensional representations learned by predicting a single value function, the value functions of 1000 random policies, or 1000 AVFs sampled using Algorithm 1. Each panel element depicts the activation of a given feature across states, with blue/red indicating low/high activation.

Figure 3 shows the representations learned by the three methods. The features learned by VALUE resemble the value function itself (top left feature) or its negated image (bottom left feature). Coarsely speaking, these features capture the general distance to the goal but little else. The features learned by RP are of even worse quality. This is because almost all random deterministic policies cause the agent to avoid the goal (appendix, Figure 12). The representation learned by AVF, on the other hand, captures the structure of the domain, including paths between distal states and focal points corresponding to rooms or parts of rooms.

Although our focus is on the use of AVFs as auxiliary tasks to a deep network, we observe the same results when discovering a representation using singular value decomposition (Section 3.2), as described in Appendix I. All in all, our results illustrate that, among all value functions, AVFs are particularly useful auxiliary tasks for representation learning.

4.3 Learning the Optimal Policy

In a final set of experiments, we consider learning a reward-maximizing policy using a pretrained representation and a model-based version of the SARSA algorithm (Rummery and Niranjan, 1994; Sutton and Barto, 1998). We compare the value-based and AVF-based representations from the previous section (VALUE and AVF), and also proto-value functions (PVF; details in Appendix H.3).

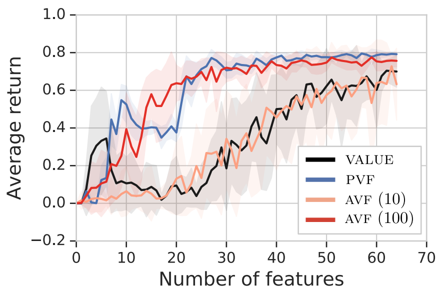

Figure 4: Average discounted return achieved by policies learned using a representation produced by VALUE, AVF, or PVF. Average is over 20 random seeds and shading gives standard deviation.

We report the quality of the learned policies after training, as a function of d, the size of the representation. Our quality measures is the average return from the designated start state (bottom left). Results are provided in Figure 4 and Figure 13 (appendix). We observe a failure of the VALUE representation to provide a useful basis for learning a good policy, even as d increases; while the representation is not rank-deficient, the features do not help reduce the approximation error.

In comparison, our AVF representations perform similarly to PVFs. Increasing the number of auxiliary tasks also leads to better representations; recall that PVF implicitly uses $n = 104$ auxiliary tasks.

5 Related Work

Our work takes inspiration from earlier research in basis or feature construction for reinforcement learning. Ratitch and Precup (2004), Foster and Dayan (2002), Menache et al. (2005), Yu and Bertsekas (2009), Bhatnagar et al. (2013), and Song et al. (2016) consider methods for adapting parametrized basis functions using iterative schemes. Including Mahadevan and Maggioni (2007)'s proto-value functions, a number of works (we note Dayan, 1993; Petrik, 2007; Mahadevan and Liu, 2010; Ruan et al., 2015; Barreto et al., 2017) have used characteristics of the transition structure of the MDP to generate representations; these are the closest in spirit to our approach, although none use the reward or consider the geometry of the space of value functions. Parr et al. (2007) proposed constructing a representation from successive Bellman errors, Keller et al. (2006) used dimensionality reduction methods; finally Hutter (2009) proposes a universal scheme for selecting representations.

Deep reinforcement learning algorithms have made extensive use of auxiliary tasks to improve agent performance, beginning perhaps with universal value function approximators (Schaul et al., 2015) and the UNREAL architecture (Jaderberg et al., 2017); see also Dosovitskiy and Koltun (2017), François-Lavet et al. (2018) and, more tangentially, van den Oord et al. (2018). Levine et al. (2017) and Chung et al. (2019) make explicit use of two-part approximations to derive more sample efficient deep reinforcement learning algorithms. The notion of augmenting an agent with side predictions regarding the world is not new, with roots in TD models (Sutton, 1995), predictive state representations (Littman et al., 2002) and the Horde architecture (Sutton et al., 2011), which itself is based on Selfridge (1959)'s Pandemonium architecture.

In passing, we remark on a number of works which aim to quantify or explain the usefulness of a representation. Parr et al. (2008) studies the particular case of linear representations, while Li et al. (2006); Abel et al. (2016) consider the approximation error that arises from state abstraction. More recently, Nachum et al. (2019) provide some interesting guarantees in the context of hierarchical reinforcement learning, while Such et al. (2019) visualizes the representations learned by Atari-playing agents. Finally, Bertsekas (2018) remarks on the two-part approximation we study here.

6 Conclusion

In this paper we studied the notion of an adversarial value function, derived from a geometric perspective on representation learning in RL. Our work shows that adversarial value functions exhibit interesting structure, and are good auxiliary tasks when learning a representation of an environment. We believe our work to be the first to provide formal evidence as to the usefulness of predicting value functions for shaping an agent's representation.

Our work opens up the possibility of automatically generating auxiliary tasks in deep reinforcement learning, analogous to how deep learning itself enabled a move away from hand-crafted features. A number of practical considerations remain to be addressed. First, our sampling procedure is clearly inefficient, and may be improved by encouraging diversity within AVFs. Second, practical implementations require learning the AVFs concurrently with the main task. Doing results in off-policy learning, whose negative effects are well-documented even in recent applications. (e.g. van Hasselt et al., 2018). Third, interest functions in large domains should incorporate some degree of smoothness, rather than vary rapidly from state to state.

From a mathematical perspective, our formulation of the representation learning problem (1) was made with both convenience and geometry in mind. Conceptually, it may be interesting to consider our approach in other norms, including the weighted norms used in approximation results.

7 Acknowledgements

The authors thank the many people who helped shape this project through discussions and feedback on early drafts: Lihong Li, George Tucker, Doina Precup, Ofir Nachum, Csaba Szepesvári, Georg Ostrovski, Marek Petrik, Marlos Machado, Tim Lillicrap, Danny Tarlow, Saurabh Kumar, and Carles

Gelada. Special thanks also to Philip Thomas and Scott Niekum, who gave this project its initial impetus.

8 Author Contributions

M.G.B., W.D., D.S., and N.L.R. conceptualized the representation learning problem. M.G.B., W.D., T.L., A.A.T., R.D., D.S., and N.L.R. contributed to the theoretical results. M.G.B., W.D., P.S.C., R.D., and C.L. performed experiments and collated results. All authors contributed to the writing.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *Symposium on Operating Systems Design and Implementation*.

Abel, D., Hershkowitz, D. E., and Littman, M. L. (2016). Near optimal behavior via approximate state abstraction. In *Proceedings of the International Conference on Machine Learning*.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in Neural Information Processing Systems*.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. (2017). Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*.

Behzadian, B. and Petrik, M. (2018). Feature selection by singular value decomposition for reinforcement learning. In *Proceedings of the ICML Prediction and Generative Modeling Workshop*.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press, Princeton, NJ.

Bernhard, K. and Vygen, J. (2008). Combinatorial optimization: Theory and algorithms. *Springer, Third Edition, 2005.*

Bertsekas, D. P. (2012). *Dynamic Programming and Optimal Control, Vol. II: Approximate Dynamic Programming*. Athena Scientific.

Bertsekas, D. P. (2018). Feature-based aggregation and deep reinforcement learning: A survey and some new implementations. Technical report, MIT/LIDS.

Bhatnagar, S., Borkar, V. S., and Prabuchandran, K. (2013). Feature search in the Grassmanian in online reinforcement learning. *IEEE Journal of Selected Topics in Signal Processing*.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. (2018). Dopamine: A research framework for deep reinforcement learning. *arXiv*.

Chung, W., Nath, S., Joseph, A. G., and White, M. (2019). Two-timescale networks for nonlinear value function approximation. In *International Conference on Learning Representations*.

Dadashi, R., Taïga, A. A., Roux, N. L., Schuurmans, D., and Bellemare, M. G. (2019). The value function polytope in reinforcement learning. *arXiv*.

Dayan, P. (1993). Improving generalisation for temporal difference learning: The successor representation. *Neural Computation*.

Dosovitskiy, A. and Koltun, V. (2017). Learning to act by predicting the future. In *Proceedings of the International Conference on Learning Representations*.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556.

Farahmand, A., Ghavamzadeh, M., Szepesvári, C., and Mannor, S. (2016). Regularized policy iteration with nonparametric function spaces. *Journal of Machine Learning Research*.

Foster, D. and Dayan, P. (2002). Structure in the space of value functions. *Machine Learning*.

François-Lavet, V., Bengio, Y., Precup, D., and Pineau, J. (2018). Combined reinforcement learning via abstract representations. *arXiv*.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. (2019). DeepMDP: Learning continuous latent space modelsfor representation learning. In *Proceedings of the International Conference on Machine Learning*.

Hutter, M. (2009). Feature reinforcement learning: Part I. Unstructured MDPs. *Journal of Artificial General Intelligence*.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2017). Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of the International Conference on Learning Representations*.

Keller, P. W., Mannor, S., and Precup, D. (2006). Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Lagoudakis, M. and Parr, R. (2003). Least-squares policy iteration. *The Journal of Machine Learning Research*.

Levine, N., Zahavy, T., Mankowitz, D., Tamar, A., and Mannor, S. (2017). Shallow updates for deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Li, L., Walsh, T., and Littman, M. (2006). Towards a unified theory of state abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*.

Liang, Y., Machado, M. C., Talvitie, E., and Bowling, M. H. (2016). State of the art control of atari games using shallow reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*.

Littman, M. L., Sutton, R. S., and Singh, S. (2002). Predictive representations of state. In *Advances in Neural Information Processing Systems*.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2017). A Laplacian framework for option discovery in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. (2018). Eigenoption discovery through the deep successor representation. In *Proceedings of the International Conference on Learning Representations*.

Mahadevan, S. and Liu, B. (2010). Basis construction from power series expansions of value functions. In *Advances in Neural Information Processing Systems*.

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

Menache, I., Mannor, S., and Shimkin, N. (2005). Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*.

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

Munos, R. (2003). Error bounds for approximate policy iteration. In *Proceedings of the International Conference on Machine Learning*.

Munos, R. (2007). Performance bounds in l_p-norm for approximate value iteration. *SIAM Journal on Control and Optimization*.

Nachum, O., Gu, S., Lee, H., and Levine, S. (2019). Near-optimal representation learning for hierarchical reinforcement learning. In *Proceedings of the International Conference on Learning Representations*.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M. L. (2008). An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Parr, R., Painter-Wakefield, C., Li, L., and Littman, M. (2007). Analyzing feature generation for value-function approximation. In *Proceedings of the International Conference on Machine Learning*.

Petrik, M. (2007). An analysis of Laplacian methods for value function approximation in MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Petrik, M. and Zilberstein, S. (2011). Robust approximate bilinear programming for value function approximation. *Journal of Machine Learning Research*.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Ratitch, B. and Precup, D. (2004). Sparse distributed memories for on-line value-based reinforcement learning. In *Proceedings of the European Conference on Machine Learning*.

Rockafellar, R. T. and Wets, R. J.-B. (2009). *Variational analysis*. Springer Science & Business Media.

Ruan, S. S., Comanici, G., Panangaden, P., and Precup, D. (2015). Representation discovery for mdps using bisimulation metrics. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical report, Cambridge University Engineering Department.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *Proceedings of the International Conference on Machine Learning*.

Selfridge, O. (1959). Pandemonium: A paradigm for learning. In *Symposium on the mechanization of thought processes*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Solway, A., Diuk, C., Córdova, N., Yee, D., Barto, A. G., Niv, Y., and Botvinick, M. M. (2014). Optimal behavioral hierarchy. *PLOS Computational Biology*.

Song, Z., Parr, R., Liao, X., and Carin, L. (2016). Linear feature encoding for reinforcement learning. In *Advances in Neural Information Processing Systems*.

Stachenfeld, K. L., Botvinick, M., and Gershman, S. J. (2014). Design principles of the hippocampal cognitive map. In *Advances in Neural Information Processing Systems*.

Such, F. P., Madhavan, V., Liu, R., Wang, R., Castro, P. S., Li, Y., Schubert, L., Bellemare, M. G., Clune, J., and Lehman, J. (2019). An Atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Sutton, R., Modayil, J., Delp, M., Degris, T., Pilarski, P., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the International Conference on Autonomous Agents and Multiagents Systems*.

Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time scales. In *Proceedings of the International Conference on Machine Learning*.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S., Mahmood, A. R., and White, M. (2016). An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*.

Sutton, R. S., Precup, D., and Singh, S. P. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3).

Tieleman, T. and Hinton, G. (2012). RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Tosatto, S., Pirotta, M., D'Eramo, C., and Restelli, M. (2017). Boosted fitted q-iteration. In *Proceedings of the International Conference on Machine Learning*.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. In *Advances in Neural Information Processing Systems*.

van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. (2018). Deep reinforcement learning and the deadly triad. *arXiv*.

Wu, Y., Tucker, G., and Nachum, O. (2019). The laplacian in rl: Learning representations with efficient approximations. In *Proceedings of the International Conference on Learning Representations (to appear)*.

Yu, H. and Bertsekas, D. P. (2009). Basis function adaptation methods for cost approximation in mdp. In *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*.

A Proof of Lemma 1

Consider the value polytope $\mathcal{V}$. We have using Corollary 1 of Dadashi et al. (2019) that

$$\mathcal{V} \subseteq \textsc{Conv}(\mathcal{V}) = \textsc{Conv}(V^{\pi_1}, \ldots, V^{\pi_m}), \tag{5}$$

where $\pi_1, \ldots, \pi_m$ is a finite collection of deterministic policies. We assume that this set of policies is of minimal cardinality e.g. the value functions $V^{\pi_1}, \ldots, V^{\pi_m}$ are distinct.

The optimization problem $\max\limits_{V \in \mathcal{V}} \delta^\top V$ is equivalent to the linear program $\max\limits_{V \in \textsc{Conv}(\mathcal{V})} \delta^\top V$, and the maximum is reached at a vertex U of the convex hull of $\mathcal{V}$ (Boyd and Vandenberghe, 2004). By (5), U is the value function of a deterministic policy. Now consider $\delta \in \mathbb{R}^n$ such that f_δ attains its maximum over multiple elements of the convex hull. By hypothesis, there must be two policies π_i, π_j such that $V^{\pi_i} \neq V^{\pi_j}$ and

$$\max_{V \in \mathcal{V}} \delta^\top V = \delta^\top V^{\pi_i} = \delta^\top V^{\pi_j},$$

and thus

$$\delta^\top (V^{\pi_i} - V^{\pi_j}) = 0. \tag{6}$$

Write Δ for the ensemble of such δ. We have from (6):

$$\Delta \subseteq \bigcup_{1 \le i < j \le m} \{\delta \in \mathbb{R}^n \mid \delta^T (V_{\pi_i} - V_{\pi_j}) = 0\}.$$

As $V^{\pi_1}, \ldots, V^{\pi_m}$ are distinct, Δ is included in a finite union of hyperplanes (recall that hyperplanes of $\mathbb{R}^n$ are vector spaces of dimension $n-1$). The Lebesgue measure of a hyperplane is 0 (in $\mathbb{R}^n$), hence a finite union of hyperplanes also has Lebesgue measure 0. Hence Δ itself has Lebesgue measure of 0 in $\mathbb{R}^n$.

B Proof of Corollary 2

Similarly to the proof of Lemma 1, we introduce $V^{\pi_1}, \ldots, V^{\pi_m}$ which are the distinct vertices of the convex hull of the value polytope $\mathcal{V}$. Note that $\pi_1, \ldots, \pi_m$ are deterministic policies. We shall show that there are at most 2^n such vertices.

Recall the definition of a cone in $\mathbb{R}^n$: C is a cone in $\mathbb{R}^n$ if $\forall v \in C, \forall \alpha \geq 0, \alpha v \in C$. For each vertex V^{π_i}, Rockafellar and Wets (2009, Theorem 6.12) states that there is an associated cone C_i of nonzero Lebesgue measure in $\mathbb{R}^n$ such that

$$\forall \delta \in C_i,\ \arg\max_{V \in \mathcal{V}} \delta^\top V = V^{\pi_i}.$$

Now using Theorem 2, we have

$$\max_{V \in \mathcal{V}} \delta^\top V = \max_{\pi \in \mathcal{P}} d_\pi^\top r, \text{ where } d_\pi = (I - \gamma P^{\pi\top})^{-1} \delta.$$

For all $\delta \in C_i$ the corresponding policy π_i is the same (by hypothesis). For such a δ, define $d_{\pi_i, \delta} := (I - \gamma P^{\pi_i \top})^{-1} \delta$, such that

$$\delta^\top V^{\pi_i} = d_{\pi_i, \delta}^\top r.$$

Because C_i is a cone of nonzero Lebesgue measure in $\mathbb{R}^n$, we have $span(C_i) = \mathbb{R}^n$. Combined with the fact that $(I - \gamma P^{\pi_i \top})^{-1}$ is full rank, this implies we can find a direction δ_i in C_i for which $d_{\pi_i, \delta_i}(x) \neq 0$ for all $x \in \mathcal{X}$. For this δ_i, using Theorem 2 we have:

$$V^{\pi_i}(x) = r(x) + \gamma \begin{cases} \max_{a \in \mathcal{A}} \mathbb{E}_{x' \sim P} V^{\pi_i}(x') & d_{\pi_i, \delta_i}(x) > 0, \\ \min_{a \in \mathcal{A}} \mathbb{E}_{x' \sim P} V^{\pi_i}(x') & d_{\pi_i, \delta_i}(x) < 0, \end{cases} \tag{7}$$

and each state is "strictly" a maximizer or minimizer (the purpose of our cone argument was to avoid the undefined case where $d_{\pi_i, \delta_i}(x) = 0$). Now define $\sigma_i \in \{-1, 1\}^n$, $\sigma_i(x) = \text{sign}(d_{\pi_i, \delta_i}(x))$. We have:

$$\begin{aligned} V^{\pi_i}(x) &= r(x) + \gamma \sigma_i(x) \max_{a \in \mathcal{A}} \sigma_i(x) \mathop{\mathbb{E}}_{x' \sim P} V^{\pi_i}(x') \\ &= \mathcal{T}_{\sigma_i} V^{\pi_i}(x) \end{aligned}$$

where $\mathcal{T}_\sigma V(x) = r(x) + \gamma\sigma(x)\max_{a\in\mathcal{A}}\sigma(x)\,\mathbb{E}_{x'\sim P} V(x')$ for $\sigma \in \{-1,1\}^n$. We show that $\mathcal{T}_\sigma$ is a contraction mapping: for any $x \in \mathcal{X}$ and $\sigma \in \{-1,1\}^n$,

$$\begin{aligned}
|\mathcal{T}_\sigma V_1(x) - \mathcal{T}_\sigma V_2(x)| &= |r(x) + \gamma\sigma(x)\max_{a\in\mathcal{A}}\sigma(x)\mathop{\mathbb{E}}_{x'\sim P} V_1(x') - r(x) - \gamma\sigma(x)\max_{a\in\mathcal{A}}\sigma(x)\mathop{\mathbb{E}}_{x'\sim P} V_2(x')| \\
&= \gamma|\max_{a\in\mathcal{A}}\sigma(x)\mathop{\mathbb{E}}_{x'\sim P} V_1(x') - \max_{a\in\mathcal{A}}\sigma(x)\mathop{\mathbb{E}}_{x'\sim P} V_2(x')| \\
&\le \gamma\max_{a\in\mathcal{A}}|\sigma(x)\mathop{\mathbb{E}}_{x'\sim P} V_1(x') - \sigma(x)\mathop{\mathbb{E}}_{x'\sim P} V_2(x')| \\
&\le \gamma\max_{a\in\mathcal{A}}\max_{x'\in\mathcal{X}}|V_1(x') - V_2(x')| \\
&= \gamma\max_{x'\in\mathcal{X}}|V_1(x') - V_2(x')|.
\end{aligned}$$

Therefore, $\|\mathcal{T}_\sigma V_1 - \mathcal{T}_\sigma V_2\|_\infty \le \gamma\|V_1 - V_2\|_\infty$ and $\mathcal{T}_\sigma$ is a γ-contraction in the supremum norm. By Banach's fixed point theorem V^{π_i} is its a unique fixed point.

We showed that each vertex V^{π_i} of the value function polytope $\mathcal{V}$ is the fixed point of an operator $\mathcal{T}_{\sigma_i}$. Since there are 2^n such operators, there are at most 2^n vertices.

C Proof of Theorem 1

We will show that the maximization over $\mathcal{P}$ is the same as the maximization over $\mathcal{P}_v$.

Let Π_ϕ be the projection matrix onto the hyperplane H spanned by the basis induced by ϕ. We write

$$\begin{aligned}
\|\hat{V}^\pi_\phi - V^\pi\|_2^2 &= \|\Pi_\phi V^\pi - V^\pi\|_2^2 \\
&= \|(\Pi_\phi - I)V^\pi\|_2^2 \\
&= {V^\pi}^\top(\Pi_\phi - I)^\top(\Pi_\phi - I)V^\pi \\
&= {V^\pi}^\top(I - \Pi_\phi)V^\pi
\end{aligned}$$

because Π_ϕ is idempotent. The eigenvalues of $A = I - \Pi_\phi$ are 1 and 0, and the eigenvectors corresponding to eigenvalue 1 are normal to H. Because we are otherwise free to choose any basis spanning the subspace normal to H, there is a unit vector δ normal to H for which

$$\begin{aligned}
\max_\pi \|\hat{V}^\pi_\phi - V^\pi\|_2^2 &= \max_{V^\pi\in\mathcal{V}} \|\hat{V}^\pi_\phi - V^\pi\|_2^2 \\
&= \max_{V^\pi\in\mathcal{V}} {V^\pi}^\top\delta\delta^\top V^\pi.
\end{aligned}$$

Denote the value function maximizing this quantity by V^π_{MAX}. This δ can be chosen so that $\delta^\top V^\pi_{\text{MAX}} > 0$ (if not, take $\delta' = -\delta$). Then V^π_{MAX} is also the maximizer of $f(V) := \delta^\top V$ over $\mathcal{V}$, and Lemma 1 tells us that V^π_{MAX} is an extremal vertex.

D Proof of Theorem 2

To begin, note that

$$\begin{aligned}
\delta^\top V^\pi &= \delta^\top(I - \gamma P^\pi)^{-1} r \\
&= (I - \gamma {P^\pi}^\top)^{-1}\delta^\top r \\
&= d_\pi^\top r,
\end{aligned}$$

as required.

Now, we choose an indexing for states in $\mathcal{S}$ and will refer to states by their index.

Let π be the policy maximizing $\delta^\top V^\pi$ and consider some $x^* \in \mathcal{S}$. We assume without loss of generality that x^* is the first state in the previous ordering. Recall that $n = |\mathcal{S}|$.

The theorem states that policy π chooses the highest-valued action at x^* if $d_\pi(x^*) > 0$, and the lowest-valued action if $d_\pi(x^*) < 0$. Writing $P^\pi_{x^*} := P^\pi(\cdot \,|\, x^*)$ for conciseness, this is equivalent to

$$r(x^*) + \mathop{\mathbb{E}}_{x'\sim P^\pi_{x^*}} V^\pi(x') = \max_{\pi'} r(x^*) + \mathop{\mathbb{E}}_{x'\sim P^{\pi'}_{x^*}} V^\pi(x'),$$

for $d_\pi(x^*) > 0$, and conversely with a $\min_{\pi'}$ for $d_\pi(x^*) < 0$ (equivalently, $\mathcal{T}^\pi V^\pi(x^*) \geq \mathcal{T}^{\pi'} V^\pi(x^*)$ for all $\pi' \in \mathcal{P}$ or $\mathcal{T}^\pi V^\pi(x^*) \leq \mathcal{T}^{\pi'} V^\pi(x^*)$ in operator notation).

We write the transition matrix P^π as follows

$$P^\pi = \begin{pmatrix} L_1^\pi \\ \vdots \\ L_n^\pi \end{pmatrix}.$$

Where $L_i^\pi = \big(P^\pi(x_1 \,|\, x_i), \cdots, P^\pi(x_n \,|\, x_i)\big)$ is P^π's i-th row.

Then we express the transition matrix as $P^\pi = A^\pi + B^\pi$, with A^π and B^π given by

$$A^\pi = \begin{pmatrix} 0 \\ L_2^\pi \\ \vdots \\ L_n^\pi \end{pmatrix} \quad B^\pi = \begin{pmatrix} L_1^\pi \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

We can then write

$$\begin{aligned} V^\pi &= r + \gamma P^\pi V^\pi \\ &= r + \gamma (A^\pi + B^\pi) V^\pi \\ \Rightarrow \; V^\pi &= (I - \gamma A^\pi)^{-1} (r + \gamma B^\pi V^\pi). \end{aligned}$$

This is an application of matrix splitting (e.g Puterman, 1994). The invertibility of $(I - \gamma A^\pi)$ is guaranteed because A^π is a substochastic matrix. The first term of the r.h.s corresponds to the expected sum of discounted rewards when following π until reaching x^*, while the second term is the expected sum of discounted rewards received after leaving from x^* and following policy π.

Note that $(I - \gamma A^\pi)^{-1}$ does not depend on $\pi(\cdot \,|\, x^*)$ and that

$$B^\pi V^\pi = \begin{pmatrix} \mathbb{E}_{x' \sim P^\pi_{x^*}} V^\pi(x') \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Write $C^\pi = (I - \gamma A^{\pi\top})^{-1} \delta$. We have

$$\begin{aligned} \delta^\top V^\pi &= \delta^\top (I - \gamma A^\pi)^{-1} (r + \gamma B^\pi V^\pi) \\ &= C^{\pi\top} (r + \gamma B^\pi V^\pi) \\ &= C^{\pi\top} r + C^\pi(x^*) \mathop{\mathbb{E}}_{x' \sim P^\pi_{x^*}} V^\pi(x'). \end{aligned}$$

Now by assumption,

$$\delta^\top V^\pi \geq \delta^\top V^{\pi'} \tag{8}$$

for any other policy $\pi' \in \mathcal{P}$. Take π' such that $\pi'(\cdot \,|\, x) = \pi(\cdot \,|\, x)$ everywhere but x^*; then $C^\pi = C^{\pi'}$ and (8) implies that

$$C^\pi(x^*) \mathop{\mathbb{E}}_{x' \sim P^\pi_{x^*}} V^\pi(x') \geq C^\pi(x^*) \mathop{\mathbb{E}}_{x' \sim P^{\pi'}_{x^*}} V^\pi(x').$$

Hence π must pick the maximizing action in x^* if $C^\pi(x^*) > 0$, and the minimizing action if $C^\pi(x^*) < 0$.

To conclude the proof, we show that $d_\pi(x^*)$ and $C^\pi(x^*)$ have the same sign. We write

$$d_\pi = \delta + \gamma (A^{\pi\top} + B^{\pi\top}) d_\pi.$$

Then

$$\begin{aligned}
(I - \gamma A^{\pi\top}) d_\pi &= \delta + \gamma B^{\pi\top} d_\pi \\
\Rightarrow \quad d_\pi &= C^\pi + \gamma (I - \gamma A^{\pi\top})^{-1} B^{\pi\top} d_\pi \\
&= \sum_{k=0}^{\infty} (\gamma (I - \gamma A^{\pi\top})^{-1} B^{\pi\top})^k C^\pi \\
&= \sum_{k=0}^{\infty} \gamma^k (D^{\pi\top})^k C^\pi .
\end{aligned}$$

Where $D^\pi = B^\pi (I - \gamma A^\pi)^{-1}$ is a matrix with non-negative components. Because B^π is sparse every row of $(D^\pi)^k$ is null except for the first one. We can write

$$(D^{\pi k})^\top = \begin{pmatrix} d_{11}^k \, 0 \cdots 0 \\ \vdots \\ d_{1n}^k \, 0 \cdots 0 \end{pmatrix} \qquad \forall i, \; d_{1i}^k \geq 0.$$

And

$$d_\pi(x^*) = \big(\sum_{k=0}^{\infty} \gamma^k d_{11}^k \big) \, C^\pi(x^*).$$

Hence $C^\pi(x^*)$ and $d_\pi(x^*)$ have the same sign.

E Proof of Theorem 3

We first transform (3) in a equivalent problem. Let $V \in \mathbb{R}^n$, and denote by $\hat{V}_\phi := \Pi_\phi V$ the orthogonal projection of V onto the subspace spanned by the columns of Φ. From Pythagoras' theorem we have, for any $V \in \mathbb{R}^n$

$$\|V\|_2^2 = \|\hat{V}_\phi - V\|_2^2 + \|\hat{V}_\phi\|_2^2$$

Then

$$\begin{aligned}
\min_{\phi \in \mathscr{R}} \mathop{\mathbb{E}}_{V \sim \xi} \|\hat{V}_\phi - V\|_2^2 &= \min_{\phi \in \mathscr{R}} \mathop{\mathbb{E}}_{V \sim \xi} \big[\|V\|_2^2 - \|\Pi_\phi V\|_2^2 \big] \\
&= \max_{\phi \in \mathscr{R}} \mathop{\mathbb{E}}_{V \sim \xi} \|\Pi_\phi V\|_2^2 .
\end{aligned}$$

Let $u_1^*, \ldots, u_d^*$ the principal components defined in Theorem 3. These form an orthonormal basis. Hence $u_1^*, \ldots, u_d^*$ is equivalently a solution of

$$\begin{aligned}
\max_{\substack{u_1, \ldots, u_d \in \mathbb{R}^n \\ \text{orthonormal}}} \sum_{i=1}^{d} \mathop{\mathbb{E}}_{V \sim \xi} (u_i^\top V)_2^2 &= \max_{\substack{u_1, \ldots, u_d \in \mathbb{R}^n \\ \text{orthonormal}}} \sum_{i=1}^{d} \mathop{\mathbb{E}}_{V \sim \xi} \|u_i^\top V u_i\|_2^2 \\
&= \max_{\substack{u_1, \ldots, u_d \in \mathbb{R}^n \\ \text{orthonormal}}} \mathop{\mathbb{E}}_{V \sim \xi} \| \sum_{i=1}^{d} u_i^\top V u_i \|_2^2 \\
&= \max_{\substack{\Phi = [u_1, \ldots, u_d] \\ \Phi^\top \Phi = I}} \mathop{\mathbb{E}}_{V \sim \xi} \|\Phi \Phi^T V\|^2 \\
&= \max_{\phi \in \mathscr{R}} \mathop{\mathbb{E}}_{V \sim \xi} \|\Pi_\phi V\|_2^2 .
\end{aligned}$$

Which gives the desired result. The equivalence with the eigenvectors of $\mathbb{E}_\xi VV^\top$ follows from writing

$$\begin{aligned}
\mathop{\mathbb{E}}_{V \sim \xi} (u^\top V)_2^2 &= \mathop{\mathbb{E}}_{V \sim \xi} u^\top V V^\top u \\
&= u^\top \mathop{\mathbb{E}}_{V \sim \xi} \big[V V^\top \big] u
\end{aligned}$$

and appealing to a Rayleigh quotient argument, since we require u_i^* to be of unit norm.

F The Optimization Problem (1) as a Quadratic Program

Proposition 1. *The optimization problem (1) is equivalent to a quadratic program with quadratic constraints.*

Proof. For completeness, let n, d be the number of states and features, respectively. We consider representations $\Phi \in \mathbb{R}^{n \times d}$. Recall that Π_ϕ is the projection operator onto the subspace spanned by Φ, that is

$$\Pi_\phi = \Phi\big(\Phi^\top \Phi\big)^{-1}\Phi^\top.$$

We will also write $\mathcal{P}_d$ for the space of deterministic policies. We write (1) in epigraph form (Boyd and Vandenberghe, 2004):

$$\begin{aligned} \text{min. } \max_{\pi} \big\|\Pi_\phi V^\pi - V^\pi\big\|_2^2 &\Leftrightarrow \\ \text{min. } \max_{\pi \in \mathcal{P}_d} \big\|\Pi_\phi V^\pi - V^\pi\big\|_2^2 &\Leftrightarrow \\ \text{min. } t \quad \text{s.t.} \big\|\Pi_\phi V^\pi - V^\pi\big\|_2^2 &\le t \;\forall \pi \in \mathcal{P}_d. \end{aligned}$$

The first equivalence comes from the fact that the extremal vertices of our polytope are achieved by deterministic policies. The norm in the constraint can be written as

$$\begin{aligned} \big\|\Pi_\phi V^\pi - V^\pi\big\|_2^2 &= \big\|(\Pi_\phi - I)V^\pi\big\|_2^2 \\ &= {V^\pi}^\top (\Pi_\phi - I)^\top (\Pi_\phi - I) V^\pi \\ &= {V^\pi}^\top (\Pi_\phi - I)^\top (\Pi_\phi - I) V^\pi \\ &\overset{(a)}{=} {V^\pi}^\top (\Pi_\phi^2 - 2\Pi_\phi + I) V^\pi \\ &\overset{(b)}{=} {V^\pi}^\top (I - \Pi_\phi) V^\pi, \end{aligned}$$

where (a) and (b) follow from the idempotency of Π_ϕ. This is

$$\big\|\Pi_\phi V^\pi - V^\pi\big\|_2^2 = {V^\pi}^\top \big(I - \Phi(\Phi^\top\Phi)^{-1}\Phi^\top\big) V^\pi.$$

To make the constraint quadratic, we further require that the representation be left-orthogonal: $\Phi^\top\Phi = I$. Hence the optimization problem (1) is equivalent to

$$\begin{gathered} \text{minimize } \; t \quad \text{s.t.} \\ {V^\pi}^\top (I - \Phi\Phi^\top) V^\pi \le t \quad \forall \pi \in \mathcal{P}_d \\ \Phi^\top \Phi = I. \end{gathered}$$

From inspection, these constraints are quadratic. □

However, there are an exponential number of deterministic policies and hence, an exponential number of constraints in our optimization problem.

G NP-hardness of Finding AVFs

Proposition 2. *Finding* $\max_{\pi \in \mathcal{P}_d} \delta^\top V^\pi$ *is NP-hard, where the input is a deterministic MDP with binary-valued reward function, discount rate* $\gamma = 1/2$ *and* $\delta : \mathcal{X} \to \{-1/4, 0, 1\}$.

We use a reduction from the optimization version of minimum set cover, which is known to be NP-hard (Bernhard and Vygen, 2008, Corollary 15.24). Let n and m be natural numbers. An instance of set cover is a collection of sets $\mathcal{C} = \{C_1, \dots, C_m\}$ where $C_i \subseteq [n] = \{1, 2, \dots, n\}$ for all $i \in [m]$. The minimum set cover problem is

$$\min_{\mathcal{J} \subseteq [m]} \left\{ |\mathcal{J}| : \bigcup_{j \in \mathcal{J}} C_j = [n] \right\}.$$

Given a Markov decision process $\langle \mathcal{X}, \mathcal{A}, r, P, \gamma \rangle$ and function $\delta : \mathcal{X} \to [-1, 1]$ define

$$R(\pi) = \sum_{x \in \mathcal{X}} \delta(x) V^\pi(x) .$$

We are interested in the optimization problem

$$\max_{\pi \in \mathcal{P}_d} R(\pi) . \tag{9}$$

When $\delta(x) \geq 0$ for all x this corresponds to finding the usual optimal policy, which can be found efficiently using dynamic programming. The propositions claims that more generally the problem is NP-hard.

Consider an instance of set cover $\mathcal{C} = \{C_1, \ldots, C_m\}$ over universe $[n]$ with $m > 1$. Define a deterministic MDP $\langle \mathcal{X}, \mathcal{A}, r, P, \gamma \rangle$ with $\gamma = 1/2$ and $n + m + 2$ states and at most m actions. The state space is $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3$ where

$$\mathcal{X}_1 = \{u_1, \ldots, u_n\} \qquad \mathcal{X}_2 = \{v_1, \ldots, v_m\} \qquad \mathcal{X}_3 = \{g, b\} .$$

The reward function is $r(x) = \mathbb{I}_{[x=g]}$. The transition function in a deterministic MDP is characterized by a function mapping states to the set of possible next states:

$$N(x) = \bigcup_{a \in \mathcal{A}} \{x' : P(x' \mid x, a) = 1\} .$$

We use $\mathcal{C}$ to choose P as a deterministic transition function for which

$$N(x) = \begin{cases} \{x\} & \text{if } x \in \mathcal{X}_3 \\ \{g, b\} & \text{if } x \in \mathcal{X}_2 \\ \{v_j : i \in C_j\} & \text{if } x = u_i \in \mathcal{X}_1 . \end{cases}$$

This means the states in $\mathcal{X}_3$ are self transitioning and states in $\mathcal{X}_2$ have transitions leading to either state in $\mathcal{X}_3$. States in $\mathcal{X}_1$ transition to states in $\mathcal{X}_2$ in a way that depends on the set cover instance. The situation is illustrated in Figure 5. Since both policies and the MDP are deterministic, we can represent a policy as a function $\pi : \mathcal{X} \to \mathcal{X}$ for which $\pi(x) \in N(x)$ for all $x \in \mathcal{X}$. To see the connection to set cover, notice that

$$\bigcup_{v_j \in \pi(\mathcal{X}_1)} C_j = [n] , \tag{10}$$

where $\pi(\mathcal{X}_1) = \{\pi(x) : x \in \mathcal{X}_1\}$. Define

$$\delta(x) = \begin{cases} 1 & \text{if } x \in \mathcal{X}_1 \\ -1/4 & \text{if } x \in \mathcal{X}_2 \\ 0 & \text{if } x \in \mathcal{X}_3 . \end{cases}$$

Using the definition of the value function and MDP,

$$\begin{aligned} R(\pi) &= \sum_{x \in \mathcal{X}} \delta(x) V^\pi(x) \\ &= \sum_{x \in \mathcal{X}_1} V^\pi(x) - \frac{1}{4} \sum_{x \in \mathcal{X}_2} V^\pi(x) \\ &= \sum_{x \in \mathcal{X}_1} V^\pi(x) - \frac{1}{4} \sum_{x \in \mathcal{X}_2} \mathbb{I}_{[\pi(x)=g]} \\ &= \frac{1}{2} \sum_{x \in \mathcal{X}_1} \mathbb{I}_{[\pi(\pi(x))=g]} - \frac{1}{4} \sum_{x \in \mathcal{X}_2} \mathbb{I}_{[\pi(x)=g]} . \end{aligned}$$

The decomposition shows that any policy maximizing (9) must satisfy $\pi(\pi(\mathcal{X}_1)) = \{g\}$ and $\pi(\mathcal{X}_2 \setminus \pi(\mathcal{X}_1)) = \{b\}$ and for such policies

$$R(\pi) = \frac{1}{2} \left(n - \frac{1}{2} |\pi(\mathcal{X}_1)| \right) .$$

In other words, a policy maximizing (9) minimizes $|\pi(\mathcal{X}_1)|$, which by (10) corresponds to finding a minimum set cover. Rearranging shows that

$$\min_{\mathcal{J} \subseteq [m]} \left\{ |\mathcal{J}| : \bigcup_{j \in \mathcal{J}} C_j = [n] \right\} = 2n - 4 \max_{\pi \in \mathcal{P}_d} R(\pi) \, .$$

The result follows by noting this reduction is clearly polynomial time.

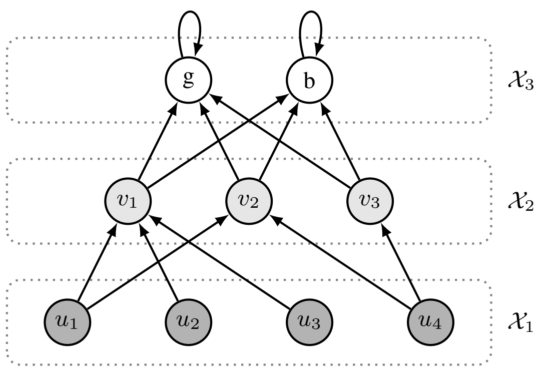

Figure 5: The challenging MDP given set cover problem $\{\{1,2,3\},\{1,4\},\{4\}\}$. State g gives a reward of 1 and all other states give reward 0. The optimal policy is to find the smallest subset of the middle layer such that for every state in the bottom layer there exists a transition to the subset.

H Empirical Studies: Methodology

H.1 Four-room Domain

The four-room domain consists of 104 discrete states arranged into four "rooms". There are four actions available to the agent, transitioning deterministically from one square to the next; when attempting to move into a wall, the agent remains in place. In our experiments, the top right state is a goal state, yielding a reward of 1 and terminating the episode; all other transitions have 0 reward.

H.2 Learning ϕ

Our representation ϕ consists of a single hidden layer of 512 rectified linear units (ReLUs) followed by a layer of d ReLUs which form our learned features. The use of ReLUs has an interesting side effect that all features are nonnegative, but other experiments with linear transforms yielded qualitatively similar results. The input is a one-hot encoding of the state (a 104-dimensional vector). All layers (and generally speaking, experiments) also included a bias unit.

The representation was learned using standard deep reinforcement learning tools taken from the Dopamine framework (Castro et al., 2018). Our loss function is the mean squared loss w.r.t. the targets, i.e. the AVFs or the usual value function. The losses were then trained using RMSProp with a step size of 0.00025 (the default optimizer from Dopamine), for 200,000 training updates each over a minibatch of size 32; empirically, we found our results robust to small changes in step sizes.

In our experiments we optimize both parts of the two-part approximation defined by ϕ and θ simultaneously, with each prediction made as a linear combination of features $\phi(x)^\top \theta_i$ and replacing $\tilde{L}(\phi; \boldsymbol{\mu})$ from (4) with a sample-based estimate. This leads to a slightly different optimization procedure but with similar representational characteristics.

H.3 Implementation Details: Proto-Value Functions

Our PVF representation consists in the top k left-singular vectors of the successor representation $(I - \gamma P^\pi)^{-1}$ for π the uniformly random policy, as suggested by Machado et al. (2018); Behzadian and Petrik (2018). See Figure 9 for an illustration.

H.4 Learning AVFs

The AVFs were learned from 1000 policy gradient steps, which were in general sufficient for convergence to an almost-deterministic policy. This policy gradient scheme was defined by directly writing the matrix $(I - \gamma P^\pi)^{-1}$ as a Tensorflow op (Abadi et al., 2016) and minimizing $-\delta^\top (I - \gamma P^\pi)^{-1} r$ w.r.t. π. We did not use an entropy penalty. In this case, there is no approximation: the AVF policies are directly represented as matrices of parameters of softmax policies.

H.5 SARSA

In early experiments we found LSPI and fitted value iteration to be somewhat unstable and eventually converged on a relatively robust, model-based variant of SARSA.

In all cases, we define the following dynamics. We maintain an occupancy vector d over the state space. At each time step we update this occupancy vector by applying one transition in the environment according to the current policy π, but also mix in a probability of resetting to a state uniformly at random in the environment:

$$d = 0.99 d P^\pi + 0.01 \mathrm{Unif}(\mathcal{X})$$

The policy itself is an ϵ-greedy policy according to the current Q-function, with $\epsilon = 0.1$.

We update the Q-function using a semi-gradient update rule based on expected SARSA (Sutton and Barto, 1998), but where we simultaneously compute updates across all states and weight them according to the occupancy d. We use a common step size of 0.01 but premultiplied the updates by the pseudoinverse of $\Phi^\top \Phi$ to deal with variable feature shapes across methods. This process was applied for 50,000 training steps, after which we report performance as the average value and/or number of steps to goal for the 10 last recorded policies (at intervals of 100 steps each).

Overall, we found this learning scheme to reduce experimental variance and to be robust to off-policy divergence, which we otherwise observed in a number of experiments involving value-only representations.

I Representations as Principal Components of Sets of Value Functions

In the main text we focused on the use of value functions as auxiliary tasks, which are combined into the representation loss (4). However, Section 3.2 shows that doing so is equivalent (in intent) to computing the principal components of a particular set of value functions, where each "column" corresponds to a particular auxiliary task.

In Figure 10 we show the representations generated from this process, using different sets of value functions. For completeness, we consider:

- 1000 AVFs,
- 1000 random deterministic policies (RDPs),
- 1000 random stochastic policies (RSPs), and
- The 104 rows of the successor matrix (corresponding to proto-value functions).

As with principal component analysis, the per-state feature activations are determined up to a signed scalar; we pick the vector which has more positive components than negative. In all but the PVF case, we sample a subset of the many possible value functions within a particular set. Figure 11 shows that the AVF approach is relatively robust to the sample size.

The AVFs are sampled using Algorithm 1, i.e. by sampling a random interest function $\delta \in [-1, 1]^n$ and using policy gradient on a softmax policy to find the corresponding value function. The random policies were generated by randomly initializing the same softmax policy and using them as-is (RSPs) or multiplying the logits by 1e6 (RDPs).

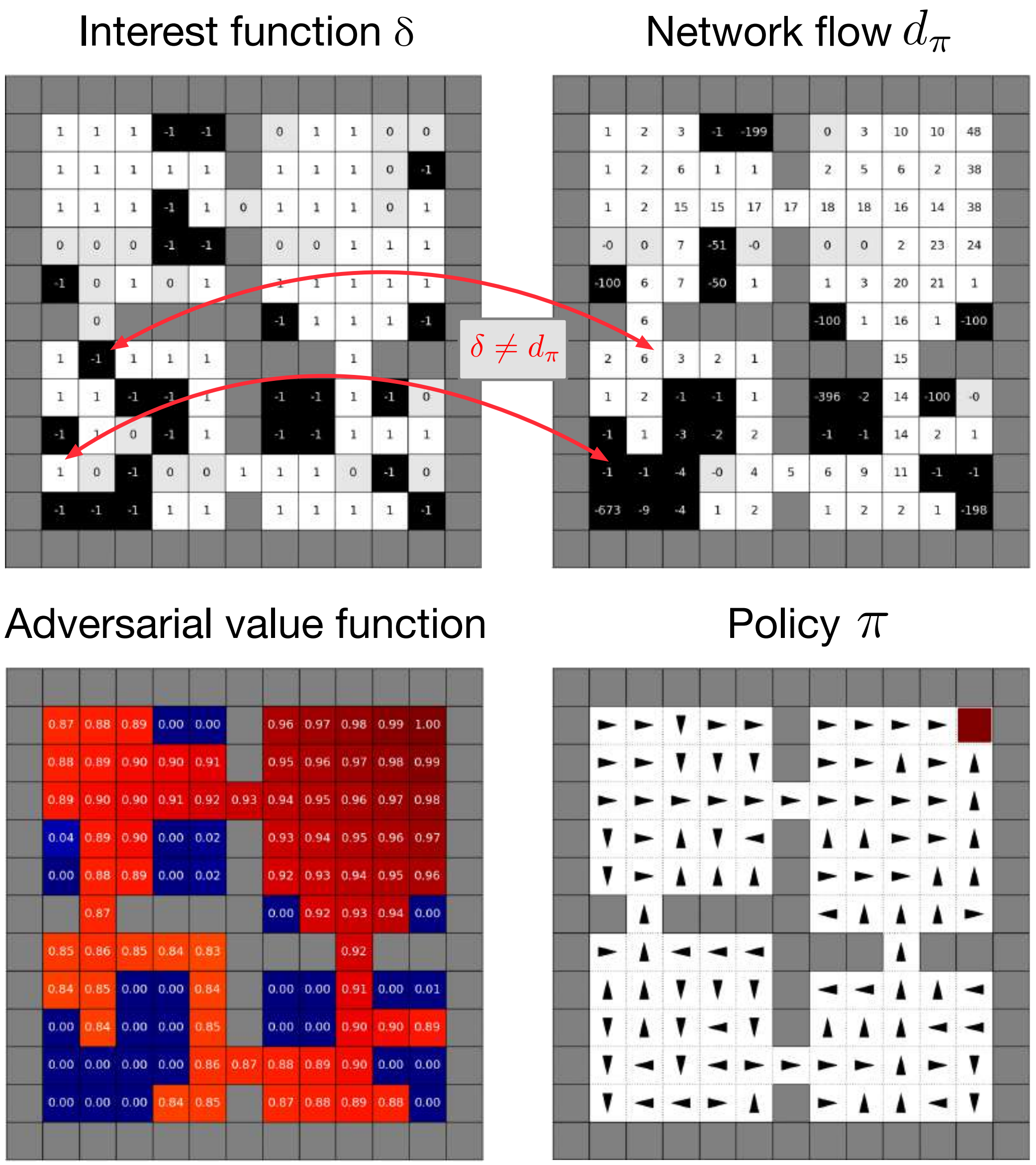

Figure 6: Figure 2, enlarged. Red arrows highlight states where δ and d_π have opposite signs.

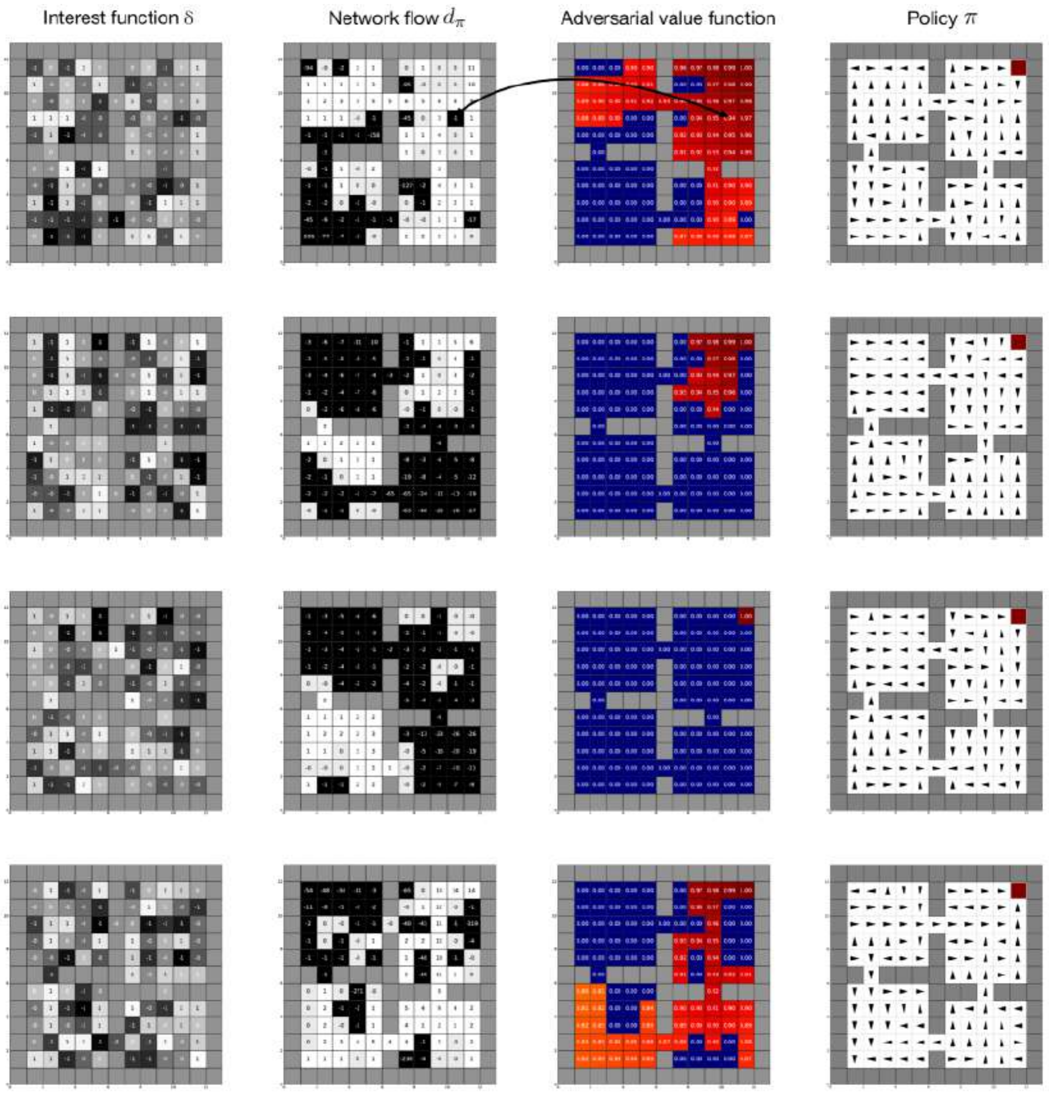

Figure 7: Four interest functions sampled from $\{-1, 1\}^n$, along with their corresponding flow d_π, adversarial value function, and corresponding policy. The top example was chosen to illustrate a scenario where $d_\pi(x) < 0$ but $V^\pi(x) > 0$; the other three were selected at random. In our experiments, sampling from $[-1, 1]^n$ yielded qualitatively similar results.

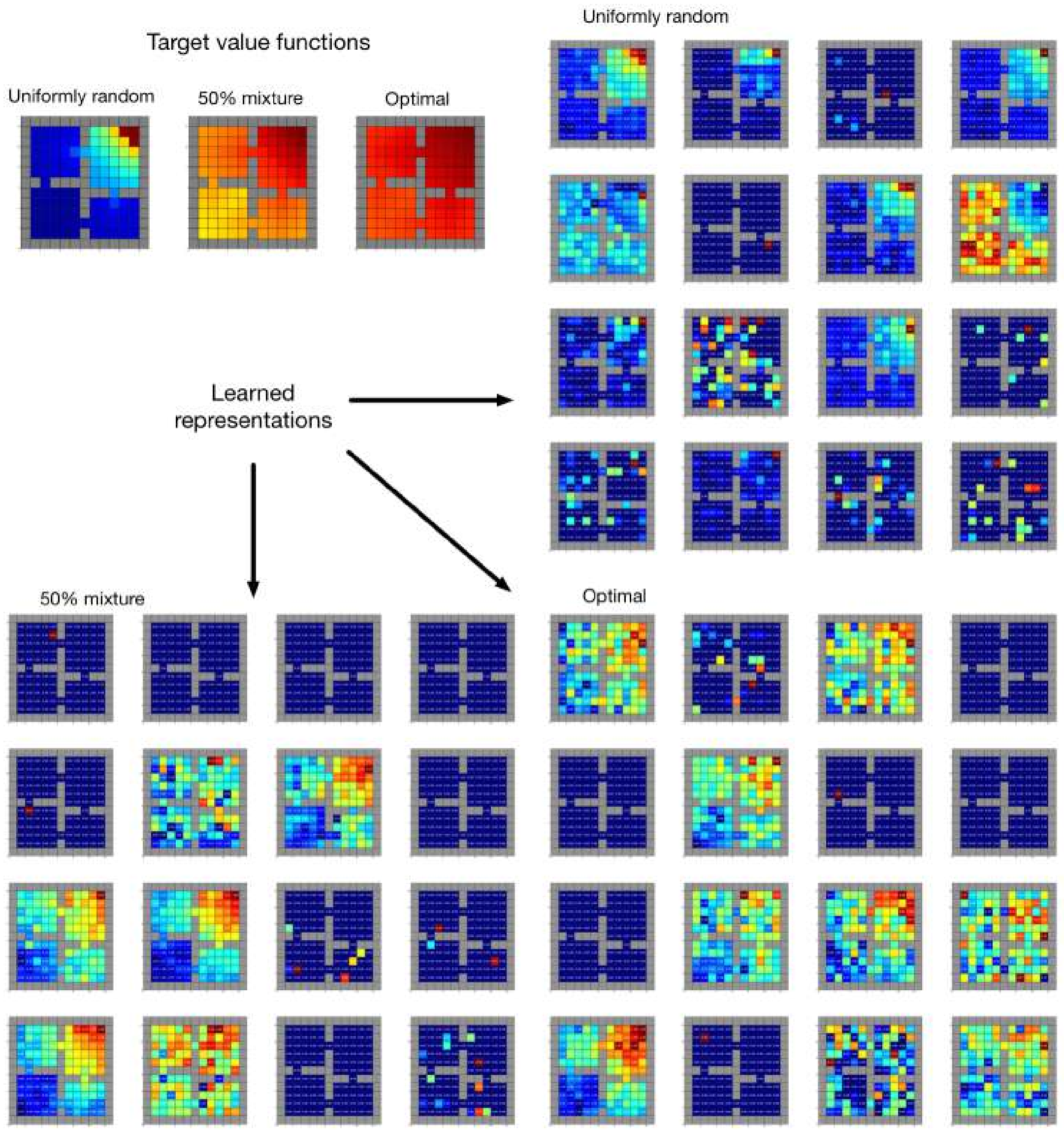

Figure 8: 16-dimensional representations learned by training a deep network to predict the value function of a single policy, namely: the uniformly random policy, the optimal policy, and a convex combination of the two in equal proportions.

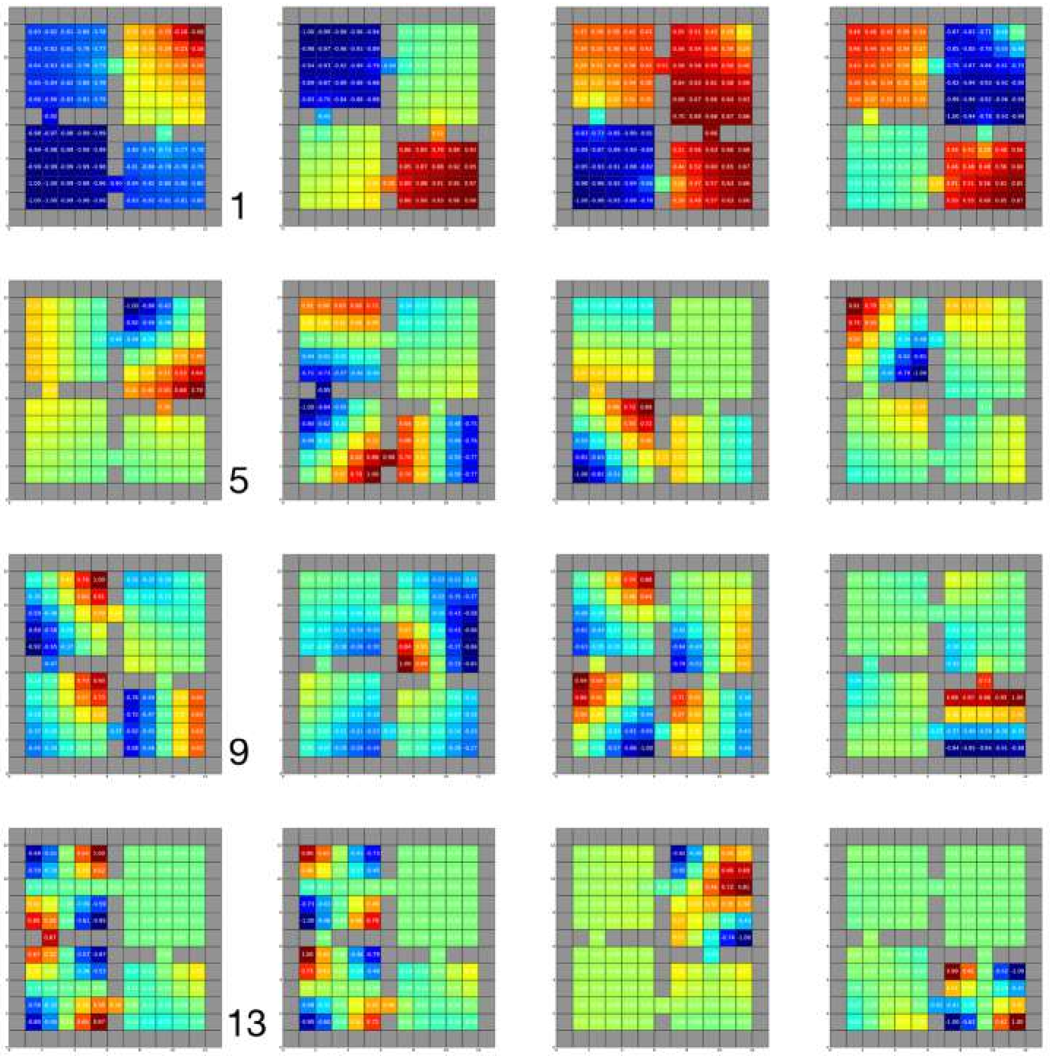

Figure 9: 16-dimensional representation generated by the proto-value function method (Mahadevan and Maggioni, 2007) applied to left-singular vectors of the transition function corresponding to the uniformly random policy. The top-left feature, labelled '1', corresponds to the second largest singular value. Notice the asymmetries arising from the absorbing goal state and the walls.

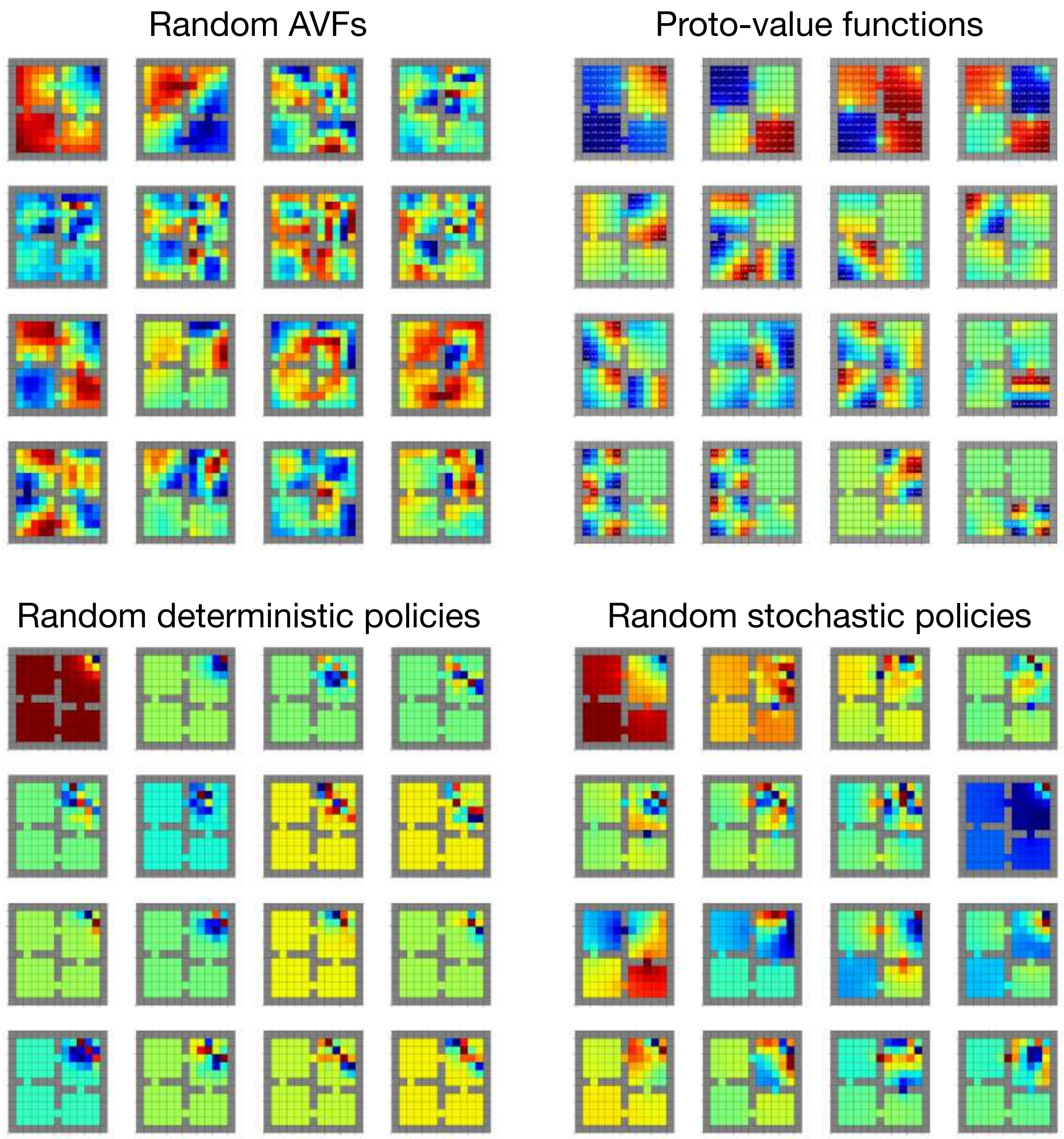

Figure 10: 16-dimensional representations generated from the principal components of different sets of value functions. Beginning in the top-left corner, in clockwise order: from $k = 1000$ AVFs sampled according as in 1; proto-value functions (9); from $k = 1000$ random deterministic policies (RDPs); and finally from $k = 1000$ random stochastic policies. Of the four, only PVFs and AVFs capture the long-range structure of the four-room domain.

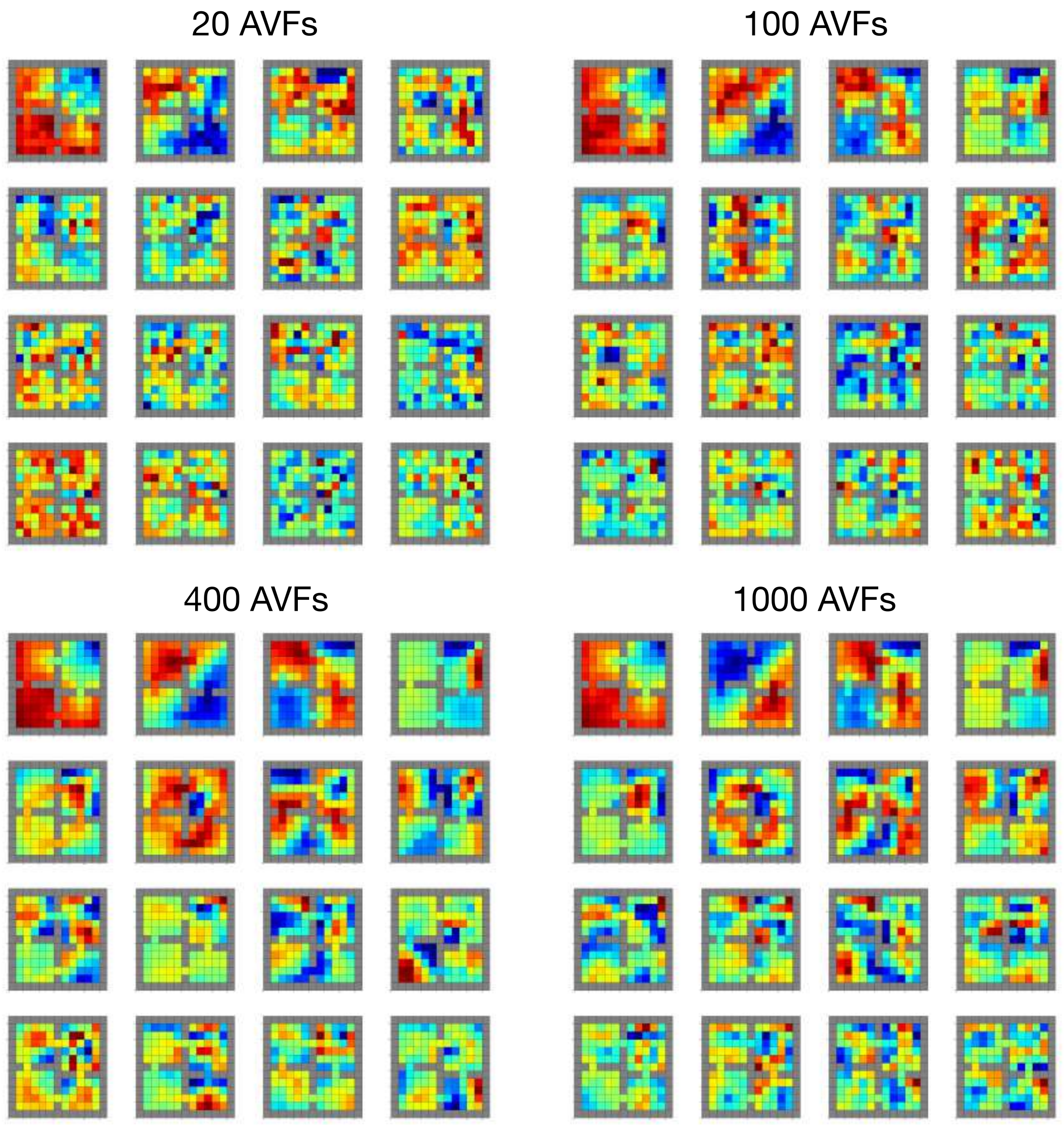

Figure 11: 16-dimensional representations generated from the principal components of sets of AVFs of varying sizes ($k = 20, 100, 400, 1000$). To minimize visualization variance, each set of AVFs contains the previous one. The accompanying video at `https://www.youtube.com/watch?v=q_XG7GhImQQ` shows the full progress from $k = 16$ to $k = 1024$.

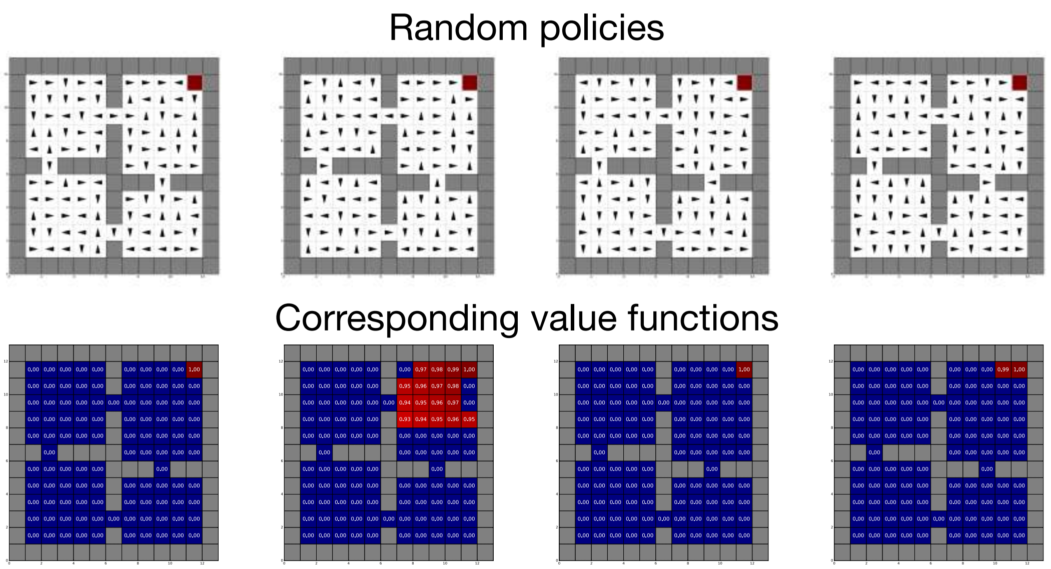

Figure 12: A sample of random deterministic policies, together with their corresponding value functions. These policies are generated by assigning a random action to each state. Under this sampling scheme, it is unlikely for a long chain of actions to reach the goal, leading to the corresponding value functions being zero almost everywhere.

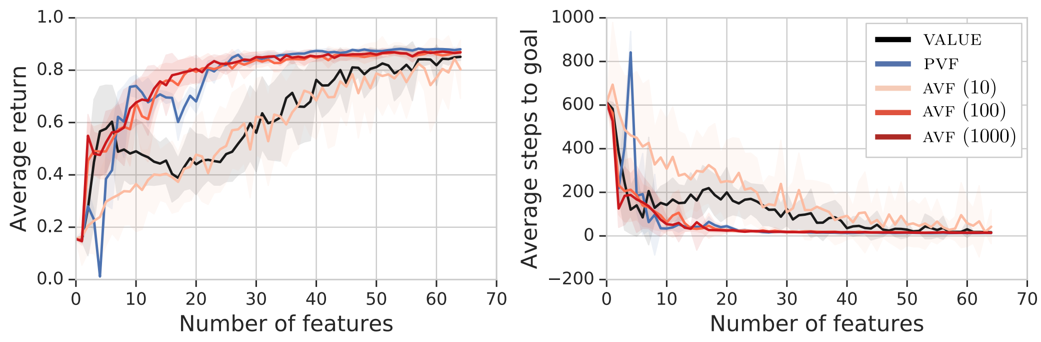

Figure 13: Average return (left) and average steps to goal (right), achieved by policies learned using a representation, with given number of features, produced by VALUE, AVF, or PVF. Average is over all states and 20 random seeds, and shading gives standard deviation.