2005

Proto-Value Functions: Developmental Reinforcement Learning

Sridhar Mahadevan
University of Massachusetts Amherst

Proto-Value Functions: Developmental Reinforcement Learning

Sridhar Mahadevan MAHADEVA@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01003

Abstract

This paper presents a novel framework called *proto-reinforcement learning* (PRL), based on a mathematical model of a *proto-value* function: these are task-independent basis functions that form the building blocks of all value functions on a given state space manifold. Proto-value functions are learned not from rewards, but instead from analyzing the *topology* of the state space. Formally, proto-value functions are Fourier eigenfunctions of the Laplace-Beltrami diffusion operator on the state space manifold. Proto-value functions facilitate structural decomposition of large state spaces, and form *geodesically* smooth orthonormal basis functions for approximating any value function. The theoretical basis for proto-value functions combines insights from spectral graph theory, harmonic analysis, and Riemannian manifolds. Proto-value functions enable a novel generation of algorithms called *representation policy iteration*, unifying the learning of representation and behavior.

1. Introduction

Reinforcement learning (RL) (Sutton & Barto, 1998) is based on the premise that value functions provide the fundamental basis for intelligent action. However, past work in this paradigm makes two assumptions: value functions are tied to task-specific rewards; also, the architecture for value function approximation is specified by a human designer, and not customized to the agent's experience of an environment. This paper addresses these shortcomings by proposing a novel framework called *proto-reinforcement learning*, based on the concept of a *proto-value function*. These are task-independent global basis functions that collectively span the space of all possible value functions on a given state space. Because they are global basis functions, they can serve as a *surrogate* value function since agents can act on the basis of linear combinations of proto-value functions. Proto-RL agents can consequently learn to act without task-specific rewards. Proto-value functions also unify three problems that face "infant" RL agents: geometric structure discovery (Menache et al., 2002; Simsek et al., 2005), representation learning, and finally, actual value function approximation incorporating *geodesic smoothing*. Proto-value functions incorporate geometric constraints intrinsic to the environment: states close in Euclidean distance may be far apart on the manifold (e.g, two states on opposite sides of a wall).

Early stages of policy learning often result in exploratory random walk behavior which generates a large sample of transitions. Proto-RL agents convert these samples into learned representations that reflect the agent's experience and an environment's *large-scale* geometry. Mathematically, the proposed framework uses a coordinate-free approach, where representations emerge from an abstract harmonic analysis of the *topology* of the underlying state space. Value functions are viewed as elements of the Hilbert space of smooth functions on a *Riemannian manifold* (Rosenberg, 1997). Hodge theory shows that the Hilbert space of smooth functions on a Riemannian manifold has a discrete spectrum captured by the eigenfunctions of the *Laplacian*, a self-adjoint operator on differentiable functions on the manifold. In the discrete setting, spectral analysis of the self-adjoint graph Laplacian operator provides an orthonormal set of basis functions for approximating any function on the graph (Chung, 1997). The graph Laplacian is an instance of a broader class of diffusion operators (Coifman & Maggioni, 2005). Proto-RL can be viewed as an *off-policy* method for representation learning: regardless of the exploration policy followed in learning the state space topology, representations emerge from a harmonic analysis of a random walk diffusion process on the state space.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

2. Proto-Value Functions

A Markov decision process (MDP) $M = \langle S, A, P^a_{ss'}, R^a_{ss'} \rangle$ is defined by a set of states S, a set of actions A, a transition model $P^a_{ss'}$ specifying the distribution over future states s' when an action a is performed in state s, and a corresponding reward model $R^a_{ss'}$ specifying a scalar cost or reward (Puterman, 1994). Abstractly, a value function is a mapping $S \rightarrow \mathcal{R}$ or equivalently a vector $\in \mathcal{R}^{|S|}$. Given a policy $\pi : S \rightarrow A$ mapping states to actions, its corresponding value function V^π specifies the expected long-term discounted sum of rewards received by the agent in any given state s when actions are chosen using the policy. Any optimal policy π^* defines the same unique optimal value function V^* which satisfies the nonlinear constraints

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma V^*(s') \right)$$

Classical techniques, such as *value iteration* and *policy iteration* (Puterman, 1994), represent value functions using an Euclidean coordinate-centered orthonormal basis $(\phi_1, \ldots, \phi_{|S|})$ for the space $\mathcal{R}^{|S|}$, where $\phi_i = [0 \ldots 1 \ldots 0]$ has a 1 only in the i^{th} position. While many methods for approximating the value function have been studied (Bertsekas & Tsitsiklis, 1996), proto-RL takes a fundamentally different *coordinate-free* approach to value function approximation based on Hilbert space theory. The notion of *operator* comes from Hilbert space theory, and forms the basis for the *coordinate-free* viewpoint. An operator is a mapping on the space of functions on the manifold (or graph). Value functions are decomposed into a linear sum of learned global basis functions constructed by spectral analysis of the *graph Laplacian* (Chung, 1997), a self-adjoint operator on the space of functions on the graph, related closely to the random walk operator. That is, a value function V^π is decomposed as

$$V^\pi = \alpha_1 V^G_1 + \ldots + \alpha_n V^G_n$$

where each V^G_i is a proto-value function defined over the state space. The basic idea is that instead of learning a task-specific value function (e.g, V^π), proto-RL agents learn the suite of proto-value functions V^G_i which form the building blocks of all value functions on the specific graph G that represents the state space.

How are proto-value functions constructed? For simplicity, assume the underlying state space is represented as an undirected graph $G = (S, E)$. The *combinatorial Laplacian* L is defined as the operator $T - A$, where T is the diagonal matrix whose entries are row sums of the adjacency matrix A. The combinatorial Laplacian L acts on any given function $f : S \rightarrow \mathcal{R}$, mapping vertices of the graph (or states) to real numbers.

$$Lf(x) = \sum_{y \sim x} (f(x) - f(y))$$

for all y adjacent to x. Consider a chain graph G consisting of a set of vertices linked in a path of length N. Given any function f on the chain graph, the combinatorial Laplacian can be viewed as a discrete analog of the well-known Laplace partial differential equation

$$\begin{aligned} Lf(v_i) &= (f(v_i) - f(v_{i-1})) + (f(v_i) - f(v_{i+1})) \\ &= (f(v_i) - f(v_{i-1})) - (f(v_{i+1}) - f(v_i)) \\ &= \nabla f(v_i, v_{i-1}) - \nabla f(v_{i+1}, f(v_i)) \\ &= \Delta f(v_i) \end{aligned}$$

Functions that solve the equation $\Delta f = 0$ are called *harmonic functions* (Axler et al., 2001). For example, on the plane $\mathcal{R}^2$, the "saddle" function $x^2 - y^2$ is harmonic. Eigenfunctions of Δ are functions f such that $\Delta f = \lambda f$, where λ is an eigenvalue of Δ. If the domain is the unit circle S^1, the trigonometric functions $\sin(\theta)$ and $\cos(\theta)$ form eigenfunctions, which leads to *Fourier* analysis.

Solving the Laplace operator on a graph means finding the eigenvalues and eigenfunctions of the equation $Lf = \lambda f$, where L is the combinatorial Laplacian computed on the graph, f is an eigenfunction, and λ is the associated eigenvalue. Later, a more sophisticated notion called the *normalized* Laplacian will be introduced, which is a symmetric self-adjoint operator that is similar to the non-symmetric random walk operator $T^{-1}A$ on a graph. The Laplacian will also be generalized to the Laplace-Beltrami operator on Riemannian manifolds. To summarize, proto-value functions are abstract Fourier basis functions that represent an orthonormal basis set for approximating any value function. Unlike trigonometric Fourier basis functions, proto-value functions or Laplacian eigenfunctions are learned from the graph topology. Consequently, they capture large-scale *geodesic* constraints, and examples of proto-value functions showing this property are shown below.

3. Examples of Proto-Value Functions

This section illustrates proto-value functions, showing their effectiveness in approximating a given value function. For simplicity, this section assumes agents have explored a given environment and constructed a complete undirected graph representing the accessibility relation between adjacent states through single-step (reversible) actions. In the next section, a complete

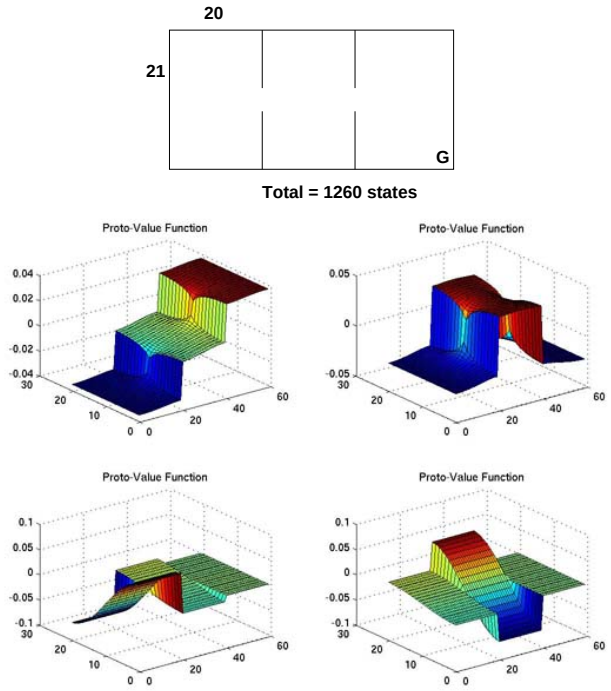

Figure 1. The proto-value functions shown are the low-order eigenfunctions of the combinatorial Laplace operator computed on the complete undirected graph representing the three room deterministic grid world environment shown. The numbers indicate the size of each room. The horizontal axes in the plots represent the length and width of the multiroom environment.

RL system is presented where graphs are learned from exploration, and then converted into basis representations that are finally used in approximating the unknown optimal value function. Note that topological learning does not require estimating probabilistic transition dynamics of actions, since representations are learned in an off-policy manner by spectral analysis of a random walk diffusion operator (the combinatorial or normalized Laplacian). Figure 1 shows proto-value functions automatically constructed from a *complete* undirected graph of a three room deterministic grid world. These basis functions capture the intrinsic *geodesic* smoothness constraints that value functions on this environment must also abide by: this synchrony is what makes them effective basis functions.

3.1. Least-Squares Approximation using Proto-Value Functions

How can proto-value functions be used to approximate a given value function? Let the basis set of proto-value functions be given by $\Phi_G = \{V_1^G, \ldots, V_k^G\}$, where each eigenfunction V_i^G is defined over all states in the neighborhood graph G on which the combinatorial Laplacian was computed (i.e, $V_i^G = (V_i^G(1), \ldots, V_i^G(|S|))$). Assume that the target value function $\hat{V}^\pi = (\hat{V}^\pi(s_1), \ldots, \hat{V}^\pi(s_m))^T$ is only known on a subset of states $S_G^m = \{s_1, \ldots, s_m\}$, where $S_G^m \subseteq S$. Define the Gram matrix $K_G = (\Phi_m^G)^T \Phi_m^G$, where Φ_m^G is the component wise projection of the basis proto-value functions onto the states in S_G^m, and $K_G(i,j) = \sum_k V_i^G(k) V_j^G(k)$. The coefficients that minimize the least-squares error is found by solving the equation $\alpha = K_G^{-1}(\Phi_M^G)^T \hat{V}^\pi$, where $\alpha = (\alpha_1, \ldots, \alpha_{|S_G|})$ is the vector of coefficients that minimizes the least-squares error. A more sophisticated *nonlinear* least-squares approach is possible, where the best approximation is computed from the k proto-value functions with the largest (absolute) coefficients. The results below were obtained using linear least-squares.

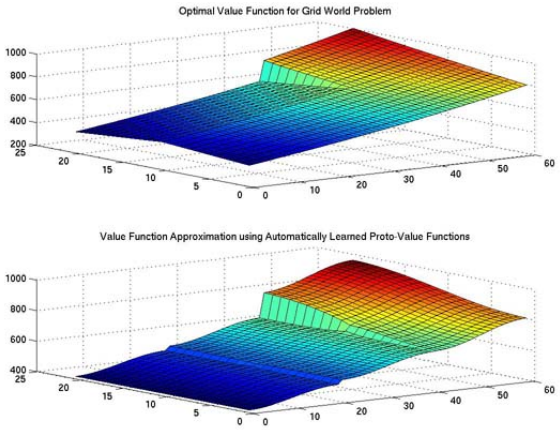

Figure 2. Proto-value functions excel at approximating value functions since they are customized to the geometry of the state space . In this figure, the value function is a vector of dimension $= \mathcal{R}^{1260}$, and is approximated by a low-dimensional $\mathcal{R}^{10}$ least-squares approximation using only 10 proto-value basis functions.

Figure 2 shows the results of linear least-squares for the three-room environment. The agent is only given a goal reward of $R = 10$ for reaching the absorbing goal state marked G in Figure 1. The discount factor $\gamma = 0.99$. Although value functions for the three-room environment are high dimensional objects in $\mathcal{R}^{1260}$, a reasonable likeness of the optimal value function is achieved using 10 proto-value functions.

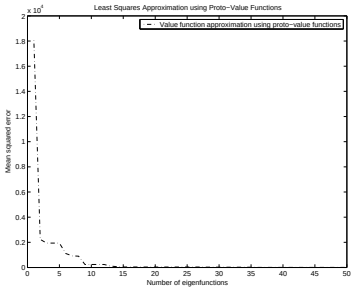

Figure 3. Mean-squared error in approximating the optimal value function for a three-room environment for varying number of basis proto-value functions.

Figure 3 plots the error in approximating the value function as the number of basis proto-value functions is increased. With 20 basis functions, the high-dimensional value function vector is fairly accurately reconstructed. To simulate value function approximation under more challenging conditions based on partial noisy samples, we generated a set of noisy samples, and compared the approximated function with the optimal value function. Figure 4 shows the results for a two-room grid world of 80 states, where noisy samples were filled in for about 17% of the states (each noisy sample was scaled by a Gaussian noise term whose mean was 1 and variance 0.1). As Figure 4 shows, the distinct character of the optimal value function is captured even with very few noisy samples.

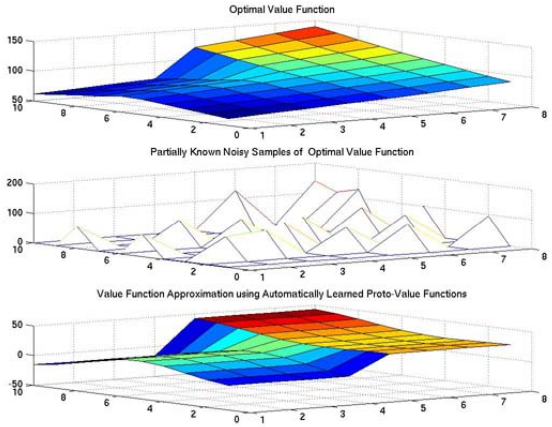

Figure 4. Proto-value function approximation (bottom plot) using 5 basis functions from a noisy partial (18%) set of samples from the optimal value function (top plot), simulating an early stage in the process of policy learning.

Finally, Figure 5 shows that proto-value functions improve on a handcoded orthogonal basis representation studied in (Koller & Parr, 2000; Lagoudakis & Parr, 2003). In this scheme, a state s is mapped to $\phi(s) = [1\ s \ldots s^i]^T$ where $i \ll |S|$. The figure compares the least mean square error with respect to the optimal (correct) value function for both the handcoded polynomial encoding and the automatically generated proto-value functions for a square grid world of size 20×20. There is a dramatic reduction in error using the learned Laplacian proto-value functions compared to the handcoded polynomial approximator. Notice how the error using polynomial approximation gets worse at higher degrees – the same behavior manifests itself below in control learning experiments.

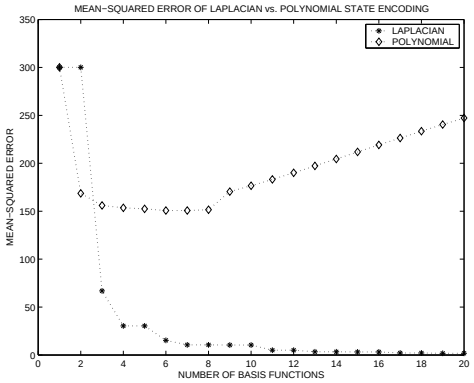

Figure 5. Mean squared error in value function approximation for a square 20×20 grid world using proto-value functions (bottom curve) versus handcoded polynomial basis functions (top curve).

4. Control Learning with Proto-RL: Representation Policy Iteration

So far, proto-RL was shown to be useful in approximating a *given* value function. We now turn to the general RL problem where agents have to learn the optimal policy by continually constructing approximations of an *unknown* optimal value function from samples of rewards. This section introduces a novel class of proto-RL algorithms called *representation policy iteration* (RPI). These methods extend the scope of Howard's classic policy iteration method (Puterman, 1994) and RL variants such as least-squares policy iteration (Lagoudakis & Parr, 2003) to learn the underlying representation for value function approximation. Our description of RPI will use LSPI as the underlying control learner, although other RL techniques such as Q-learning or SARSA could be used instead. LSPI approximates the true action-value function $Q^\pi(s, a)$ for a policy π using a set of handcoded basis functions

$\phi(s,a)$.

$$\hat{Q}^{\pi}(s,a;w) = \sum_{j=1}^{k} \phi_j(s,a) w_j$$

where the w_j are weights or parameters that can be determined using a least-squares method. Let Q^{π} be a real (column) vector $\in \mathcal{R}^{|S|\times|A|}$. The column vector $\phi(s,a)$ is a real vector of size k where each entry corresponds to the basis function $\phi_j(s,a)$ evaluated at the state action pair (s,a). The approximate action-value function can be written as $\hat{Q}^{\pi} = \Phi w^{\pi}$, where w^{π} is a real column vector of length k and Φ is a real matrix with $|S|\times|A|$ rows and k columns. Each row of Φ specifies all the basis functions for a particular state action pair (s,a), and each column represents the value of a particular basis function over all state action pairs. LSPI solves a fixed-point approximation $T_{\pi}Q^{\pi} \approx Q^{\pi}$, where T_{π} is the Bellman backup operator. This yields the following solution for the coefficients:

$$w^{\pi} = \left(\Phi^T \left(\Phi - \gamma P \Pi_{\pi} \Phi\right)\right)^{-1} \Phi^T R$$

LSPI uses the LSTDQ (least-squares TD Q-learning) method as a subroutine for learning the state-action value function $\hat{Q}^{\pi}$. The LSTDQ method solves the system of linear equations $Aw^{\pi} = b$ where

$$A = \Phi^T \Delta_{\mu} \left(\Phi - \gamma P \Pi_{\pi}\right)$$

μ is a probability distribution over $S \times A$ that defines the projection of the true action value function onto the subspace spanned by the handcoded basis functions, and $b = \Phi^T \Delta_{\mu} R$. Since A and b are unknown when learning, they are approximated from samples using the update equations

$$\begin{aligned} \tilde{A}^{t+1} &= \tilde{A}^{t} + \phi(s_t,a_t)\left(\phi(s_t,a_t) - \gamma\phi(s'_t,\pi(s'_t))\right)^T \\ \tilde{b}^{t+1} &= \tilde{b}^{t} + \phi(s_t,a_t) r_t \end{aligned}$$

where (s_t, a_t, r_t, s'_t) is the t^{th} sample of experience from a trajectory generated by the agent (using some random or guided policy). LSTDQ computes the $\tilde{A}$ matrix and $\tilde{b}$ column vector, and then returns the coefficients $\tilde{w}^{\pi}$. The overall LSPI method uses a policy iteration procedure, starting with a policy π defined by an initial weight vector w, and then repeatedly invoking LSTDQ to find the updated weights w', and terminating when the difference $\|w - w'\| \leq \epsilon$. With this brief overview of LSPI, we introduce the Representation Policy Iteration framework, which interleaves representation learning and policy learning (see Figure 6). Steps 1 and 3b automatically build customized basis functions given a set of transitions.

`Representation Policy Iteration`$(D_0, \gamma, k, \epsilon, \pi_0)$:

// D: Source of samples (s, a, r, s')
// γ: Discount factor
// ϵ: Stopping criterion
// π_o: Initial policy specified as a weight w_0.
// k: number of (unknown) basis functions

1. Use the initial source of samples D_0 to construct the basis functions $\phi_1^0, \ldots, \phi_k^0$ as follows:
   (a) Use the source of samples D_o to learn an undirected neighborhood graph G that encodes the underlying state (action) space topology.
   (b) Compute the lowest-order k eigenfunctions $\psi_1, \ldots, \psi_k$ of the (combinatorial or normalized) Laplacian on the graph G. The basis functions ϕ_i^0 for encoding state action pairs are produced by concatenating the state encoding $|A|$ times (see text for more explanation).
2. $\pi' \leftarrow \pi_0$. // $w \leftarrow w_0$
3. **repeat**
   (a) $\pi_t \leftarrow \pi'$. // $w \leftarrow w'$
   (b) **Optional**: compute a new set of basis functions ϕ^t by generating a new sample D_t by executing π_t and repeating step 1.
   (c) $\pi' \leftarrow \text{LSTDQ}(D, k, \phi^t, \gamma, \pi)$
   (d) $t \leftarrow t+1$
4. **until** $\pi \sim \pi'$ // $\|w - w'\| \leq \epsilon$

Figure 6. Representation Policy Iteration is a family of proto-reinforcement learning algorithms that learn representations and policies. Here, Least-Squares Policy Iteration is used to learn policies.

We illustrate the performance of RPI on the classic chain example from (Koller & Parr, 2000; Lagoudakis & Parr, 2003). The chain MDP, originally studied in (Koller & Parr, 2000), is a sequential open (or closed) chain of varying number of states, where there are two actions for moving left or right along the chain. The reward structure can vary, such as rewarding the agent for visiting the middle states, or the end states. Instead of using a fixed state action encoding, our approach automatically derives a customized encoding that reflects the *topology* of the chain. Figure 7 shows the basis functions that are created for an open and closed chain. Given a fixed k, the encoding $\phi(s)$ of a state s is the vector comprised of the values of the k^{th} lowest-order eigenfunctions on state k. The encoding $\phi(s,a)$ for a set of discrete actions $a \in A$ simply repeats the state encoding $|A|$ times multiplying each entry with the indicator function $I(a = a_i)$ (other schemes are of course possible).

Figure 8 shows the results of running the Representation Policy Iteration (RPI) algorithm on a 50

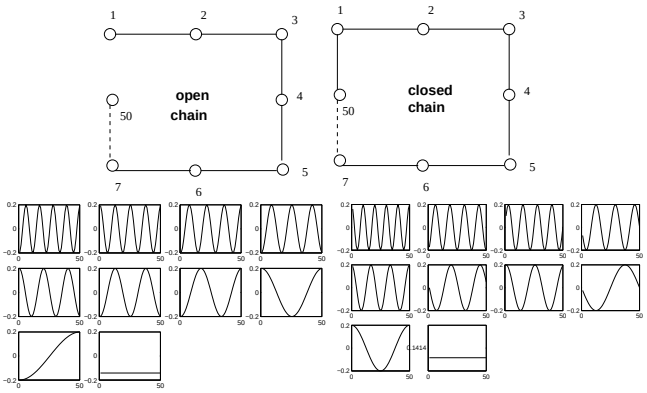

Figure 7. The first 12 orthonormal basis eigenfunctions for a 50 state open and closed chain MDP produced from a sample of 10,000 transitions by learning the underlying graph and computing its combinatorial graph Laplacian.

node chain graph, using the display format from (Lagoudakis & Parr, 2003). Here, being in states 10 and 41 earns the agent rewards of +1 and there is no reward otherwise. The optimal policy is to go right in states 1 through 9 and 26 through 41 and left in states 11 through 25 and 42 through 50. The number of samples initially collected was set at 10,000. The discount factor was set at $\gamma = 0.8$. By increasing the number of desired basis functions, it is possible to get very accurate approximation.

Table 1 compares the performance of RPI with LSPI using two handcoded basis functions studied previously with LSPI, polynomial encoding and radial-basis functions (RBF) on the 50 node chain MDP. Each row reflects the performance of either RPI using learned basis functions or LSPI with a handcoded basis function (values in parentheses indicate the number of basis functions used for each architecture). Each result is the average of five experiments on a sample of 10,000 transitions. The two numbers reported are steps to convergence and the error in the learned policy (L_1 error with respect to the optimal policy). The results show the automatically learned Laplacian basis functions in RPI provide a more stable performance at both the low end (5 basis functions) and at the higher end with $k = 25$ basis functions. As the number of basis functions are increased, RPI takes longer to converge, but learns a more accurate policy. LSPI with RBF is unstable at the low end, converging to a very poor policy for 6 basis functions. LSPI with a 5 degree polynomial approximator works reasonably well, but its performance noticeably degrades at higher degrees, converging to a very poor policy in one step for $k = 15$ and $k = 25$.

Figure 9 shows the value function learned using RPI

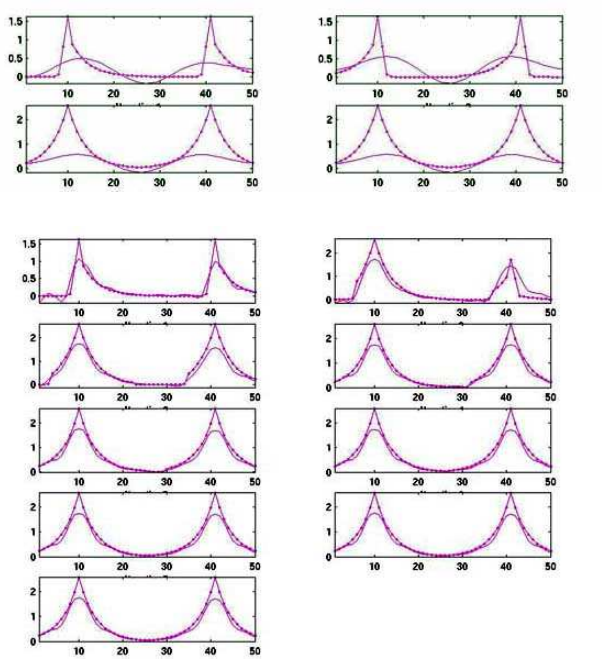

Figure 8. Representation Policy Iteration on a 50 node chain graph, for $k = 5$ basis functions (top four plots) and $k = 20$ (bottom nine plots). Each group of plots shows the value function for each iteration (numbered row wise for each group) over the 50 states. The solid curve is the approximation and the dotted curve specifies the exact function.

for a 100 state grid world domain. The actions (four compass directions) succeed with probability 0.9, and leave the agent's state unchanged otherwise. For this experiment, 5 proto-value functions were computed from the combinatorial Laplacian of an undirected graph, which was constructed from an experience sample of 18167 steps. The discount factor was set at 0.8. The agent was rewarded 100 for reaching the goal state (diagonal opposite corner in plot).

5. Theoretical Background

The theoretical basis for proto-value functions is given in this section. The Laplace-Beltrami operator is introduced in the general setting of Riemannian manifolds (Rosenberg, 1997), which motivates the discrete setting of spectral graph theory (Chung, 1997). Formally, a *manifold* $\mathcal{M}$ is a *locally Euclidean* set, with a *homeomorphism* (a bijective or one-to-one and onto mapping) from any open set containing an element $p \in \mathcal{M}$ to the n-dimensional Euclidean space $\mathcal{R}^n$. In smooth manifolds, the homeomorphism becomes a *diffeomorphism*, or a continuous bijective mapping with a continuous inverse mapping, to the Euclidean

Method	#Trials	Error
RPI (5)	4.2	-3.8
RPI (15)	7.2	-3
RPI (25)	9.4	-2
RBPF LSPI (6)	3.8	-20.8
RBPF LSPI (14)	4.4	-2.8
RBPF LSPI (26)	6.4	-2.8
Poly LSPI (5)	4.2	-4
Poly LSPI (15)	1	-34.4
Poly LSPI (25)	1	-36

Table 1. This table compares the performance of RPI using proto-value functions with LSPI using handcoded polynomial and radial basis functions on a 50 state chain graph problem. See text for explanation.

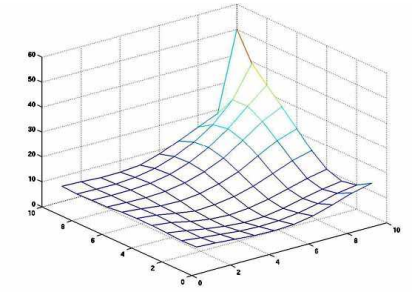

Figure 9. Value function learned after 10 iterations using Representation Policy Iteration on a 100 state gridworld MDP using 5 learned basis functions.

space $\mathcal{R}^n$. *Riemannian* manifolds are smooth manifolds where the Riemann metric defines the notion of length. Given any element $p \in \mathcal{M}$, the *tangent space* $T_p(\mathcal{M})$ is an n-dimensional vector space that is isomorphic to $\mathcal{R}^n$. A Riemannian manifold is a smooth manifold $\mathcal{M}$ with a family of smoothly varying positive definite inner products $g_p, p \in \mathcal{M}$ where $g_p : T_p(\mathcal{M}) \times T_p(\mathcal{M}) \rightarrow \mathcal{R}$. For the Euclidean space $\mathcal{R}^n$, the tangent space $T_p(\mathcal{M})$ is clearly isomorphic to $\mathcal{R}^n$ itself. One example of a Riemannian inner product on $\mathcal{R}^n$ is simply $g(x, y) = \langle x, y \rangle_{\mathcal{R}^n} = \sum_i x_i y_i$, which remains the same over the entire space.

Hodge's theorem states that any smooth function on a compact manifold has a discrete spectrum mirrored by the *eigenfunctions* of Δ, the Laplace-Beltrami self-adjoint operator. The *eigenfunctions* of Δ are functions f such that $\Delta f = \lambda f$, where λ is an eigenvalue of Δ. The *smoothness functional* for an arbitrary real-valued function on the manifold $f : \mathcal{M} \rightarrow \mathcal{R}$ is given by

$$S(f) \equiv \int_{\mathcal{M}} \|\nabla f\|^2 d\mu = \int_{\mathcal{M}} f \Delta f d\mu =< \Delta f, f >_{\mathcal{L}^2(\mathcal{M})}$$

where $\mathcal{L}_2(\mathcal{M})$ is the space of smooth functions on $\mathcal{M}$, and ∇f is the gradient vector field of f. For a Riemannian manifold $(\mathcal{M}, g)$, where the Riemannian metric g is used to define distances on manifolds, the Laplace-Beltrami operator is given as

$$\Delta = \frac{1}{\sqrt{\det\ g}} \sum_{ij} \partial_i \left(\sqrt{det\ g}\ \ g^{ij} \partial_j \right)$$

where g is the Riemannian metric, $\det g$ is the measure of volume on the manifold, and ∂_i denotes differentiation with respect to the i^{th} coordinate function.

Theorem 1 *(Hodge (Rosenberg, 1997)): Let $(\mathcal{M}, g)$ be a compact connected oriented Riemannian manifold. There exists an orthonormal basis for all smooth (square-integrable) functions $\mathcal{L}^2(\mathcal{M}, g)$ consisting of eigenfunctions of the Laplacian. All the eigenvalues are positive, except that zero is an eigenvalue with multiplicity 1.*

Hodge's theorem shows that a smooth function $f \in \mathcal{L}^2(\mathcal{M})$ can be expressed as $f(x) = \sum_{i=0}^{\infty} a_i e_i(x)$, where e_i are the eigenfunctions of Δ, i.e. $\Delta e_i = \lambda_i e_i$. The smoothness $S(e_i) =< \Delta e_i, e_i >_{\mathcal{L}^2(\mathcal{M})} = \lambda_i$.

We now turn to the discrete case. Consider an undirected graph $G = (V, E)$ without self-loops, where d_v denote the degree of vertex v. As before, define T to be the diagonal matrix where $T(v, v) = d_v$. The operator $T^{-1}A$, where A is the adjacency matrix, induces a random walk on the graph. The random walk operator is not symmetric, but it is related to a symmetric operator called the *normalized Laplacian* $\mathcal{L}$, defined as $\mathcal{L} = T^{-\frac{1}{2}} L T^{-\frac{1}{2}}$, where L is the combinatorial Laplacian. Note that $\mathcal{L} = I - T^{-\frac{1}{2}} A T^{-\frac{1}{2}}$, which implies that $T^{-1}A = T^{-\frac{1}{2}}(I - \mathcal{L})T^{\frac{1}{2}}$. In other words, the random walk operator $T^{-1}A$ is similar to $I - \mathcal{L}$ in that both have the same eigenvalues, but the eigenfunctions of the random walk operator are the eigenfunctions of $I - \mathcal{L}$ scaled by $T^{-\frac{1}{2}}$. A detailed comparison of the normalized and combinatorial Laplacian is beyond the scope of this paper, but both operators have been implemented. The *Cheeger* constant h_G of a graph G is defined as

$$h_G(S) = \min_S \frac{|E(S, \tilde{S})|}{\min(\text{vol } S, \text{vol } \tilde{S})}$$

Here, S is a subset of vertices, $\tilde{S}$ is the complement of S, and $E(S, \tilde{S})$ denotes the set of all edges (u, v) such that $u \in S$ and $v \in \tilde{S}$. The volume of a subset S is defined as $\text{vol } S = \sum_{x \in S} d_X$. The sign of the basis functions can be used to decompose state spaces (see the first proto-value function in Figure 1). Define the edge set $\partial S = \{(u, v) \in E(G) : u \in S \text{ and } v \notin S\}$. The relation between ∂S and the Cheeger constant is

given by $|\partial S| \geq h_G \text{ vol } S$. The Cheeger constant is intimately linked to the spectrum of the normalized Laplacian operator, which explains why proto-value functions capture large-scale intrinsic geometry.

Theorem 2 *(Chung, 1997): Define λ_1 to be the first (non-zero) eigenvalue of the normalized Laplacian $\mathcal{L}$ on a graph G. Let h_G denote the Cheeger constant of G. Then, we have $2h_G \geq \lambda_1$.*

6. Future Extensions

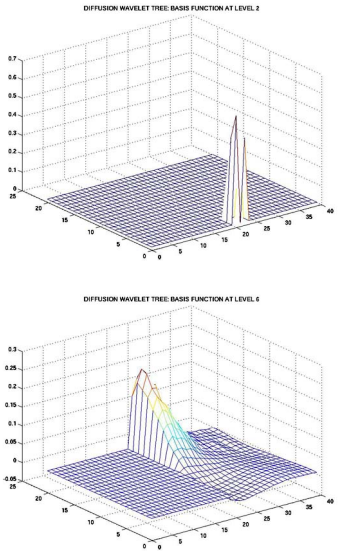

Figure 10. Diffusion wavelets are a compact multi-level representation of the Laplace-Beltrami diffusion operator. Shown here are diffusion wavelet basis functions at two levels of the hierarchy in a two-room grid world.

The Laplace-Beltrami operator is an instance of a broad class of *diffusion operators* which can be compactly represented using *diffusion wavelets* (Coifman & Maggioni, 2005) (see Figure 10). Diffusion wavelets enable fast computation of the *Green's* function or the inverse Laplacian in $O(N \log^2 N)$ time, where N is the size of the graph. A detailed investigation of diffusion wavelets for value function approximation is underway (Mahadevan & Maggioni, 2005). To enable learning representations from samples of the complete graph, *Nystrom* approximations can be used, which reduce the complexity from $O(N^3)$ to $O(m^2 N)$ where $m \ll N$ is the number of samples (Fowlkes et al., 2004). Other randomized low-rank matrix approximations are being investigated as well (Frieze et al., 1998). Another direction for scaling proto-RL is to model the state space at multiple levels of abstraction, where higher level graphs represent adjacency using temporally extended actions. Several applications of proto-RL, including high dimensional robot motion configuration planning, are ongoing.

Acknowledgments

This research was supported in part by the National Science Foundation under grant ECS-0218125. I thank Mauro Maggioni of the Department of Mathematics at Yale University for his feedback.

References

Axler, S., Bourdon, P., & Ramey, W. (2001). *Harmonic function theory*. Springer.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont, Massachusetts: Athena Scientific.

Chung, F. (1997). *Spectral Graph Theory*. American Mathematical Society.

Coifman, R., & Maggioni, M. (2005). Diffusion wavelets. *Applied Computational Harmonic Analysis*.

Fowlkes, C., Belongie, S., Chung, F., & Malik, J. (2004). Spectral grouping using the Nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 1373–1396.

Frieze, A., Kannan, R., & Vempala, S. (1998). Fast Monte-Carlo algorithms for finding low-rank approximations. *Proceedings of the IEEE Symposium on Foundations of Computer Science* (pp. 370–378).

Koller, D., & Parr, R. (2000). Policy iteration for factored MDPs. *Proceedings of the 16th Conference on Uncertainty in AI*.

Lagoudakis, M., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Mahadevan, S., & Maggioni, M. (2005). Value function approximation using diffusion wavelets and laplacian eigenfunctions. To be submitted.

Menache, I., Mannor, S., & Shimkin, N. (2002). Q-cut: Dynamic discovery of sub-goals in reinforcement learning. *ECML*.

Puterman, M. L. (1994). *Markov decision processes*. New York, USA: Wiley Interscience.

Rosenberg, S. (1997). *The Laplacian on a Riemannian Manifold*. Cambridge University Press.

Simsek, O., Wolfe, A., & Barto, A. (2005). Local graph partitioning as a basis for generating temporally extended actions in reinforcement learning. *International Conference on Machine Learning*.

Sutton, R., & Barto, A. G. (1998). *An introduction to reinforcement learning*. MIT Press.