Learning to Plan Using Harmonic Analysis of Diffusion Models

Sridhar Mahadevan, Sarah Osentoski, Jeff Johns, Kimberly Ferguson, and **Chang Wang** *
Department of Computer Science
University of Massachusetts
140 Governor's Drive
Amherst, MA 01003
(mahadeva,sosentos,johns,kferguso,chwang)@cs.umass.edu

Abstract

This paper summarizes research on a new emerging framework for learning to plan using the Markov decision process model (MDP). In this paradigm, two approaches to learning to plan have traditionally been studied: the indirect model-based approach infers the state transition matrix and reward function from samples, and then solves the Bellman equation to find the optimal (action) value function; the direct model-free approach, most notably Q-learning, estimates the action value function directly. This paper describes a new harmonic analysis framework for planning based on estimating a *diffusion model* that captures information flow on a graph (discrete state space) or a manifold (continuous state space) using the Laplace heat equation. Diffusion models are significantly easier to learn than transition models, and yet provide similar speedups in performance over model-free methods. Two methods for constructing novel plan representations from diffusion models are described: Fourier methods diagonalize a symmetric diffusion operator called the Laplacian; Wavelet methods dilate unit basis functions progressively using powers of the diffusion operator. A new variant of policy iteration – called representation policy iteration – is described consisting of an outer loop that estimates new basis functions by diagonalization or dilation, and an inner loop that learns the best policy representable within the linear span of the current basis functions. Results from continuous and discrete MDPs are provided to illustrate the new approach.

Overview of The Framework

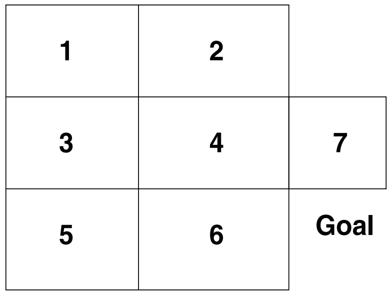
This paper summarizes research on an emerging novel framework for planning under uncertainty where both the underlying *representation* for encoding plans as well as the plans themselves are *simultaneously* learned. Two broad approaches to learning to plan can be discerned from the literature on planning using Markov decision processes (MDPs) (Puterman 1994). In the first *indirect* approach, the underlying transition matrix $P^a_{ss'}$ governing the system dynamics is learned, which describes for each action a the probability of transitioning from state s to s'. In addition, the corresponding reward function $R^a_{ss'}$, which describes the resulting payoff, is also inferred. The desired plan is then learned by solving a nonlinear system of *Bellman* equations (Puterman 1994) for the (approximately) optimal policy. In the second *direct* approach to learning to plan, the action value function $Q^\pi(s,a)$ is directly estimated from sample transitions and payoffs, using techniques from reinforcement learning (Bertsekas & Tsitsiklis 1996; Sutton & Barto 1998) such as Q-learning. In this paradigm, which is referred to as "model-free", the transition matrix never needs to be estimated.

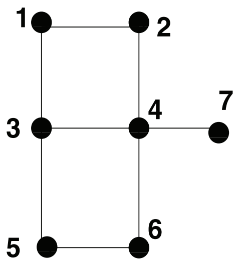
This paper provides an overview of recent research on a new *harmonic analysis* approach to learning to plan (Mahadevan 2005c; 2005a; 2005b), which is neither strictly model-based or model-free in the above sense. Harmonic analysis is a subfield of mathematics that includes *Fourier* and *wavelet* analysis. Central to the new approach is the notion of learning a *diffusion model*, which captures information flow on a graph (discrete state space) or a manifold (continuous state spaces). In the simplest setting, a diffusion model corresponds to a random walk on an undirected graph, where the edges connect states that are "adjacent" to each other. In discrete spaces, adjacency can be defined through actions, but in more general continuous spaces, adjacency can be defined through any reasonable (Euclidean) distance metric. Central to the strength of this approach is that diffusion models are considerably easier to learn than transition models: one transition from a state s to s' is sufficient to infer an edge or dependency between these two states. Figure 1 illustrates a simple example of a diffusion model.

Transition matrices combined with rewards yield value functions. Diffusion models instead are analyzed using the tools of *spectral graph theory* (Chung 1997) to yield *proto-value functions* (PVFs) (Mahadevan 2005a). Like value functions, proto-value functions map each state to a real number. Unlike value functions, proto-value functions are reward independent. It can be formally shown that any value function can be expressed as a linear combination of proto-value functions, which form an *orthogonal* basis for representing any function on a graph.

In the Fourier paradigm, PVFs are constructed by *diagonalizing* a diffusion operator, that is finding its eigenvec-

*This research was supported in part by the National Science Foundation under grants IIS-0534999.





$$\begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0.33 & 0 & 0 & 0.33 & 0.33 & 0 & 0 \end{bmatrix}$$

Figure 1: Top: A simple *diffusion model* given by an undirected unweighted graph connecting each state to neighbors that are reachable using a single (reversible) action. Bottom: first three rows of the random walk matrix $P_r = D^{-1}W$. P_r is not symmetric, but it has real eigenvalues and eigenvectors since it is spectrally similar to the symmetric normalized graph Laplacian.

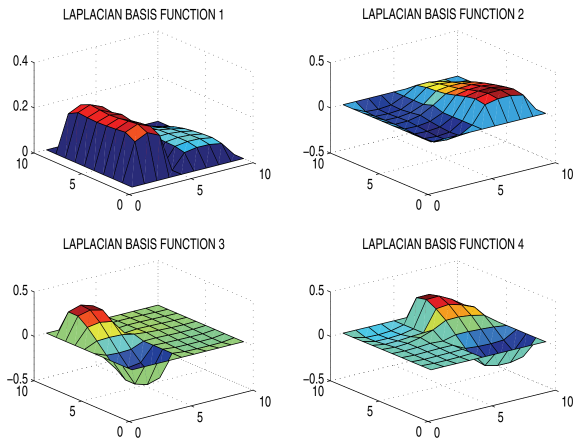


Figure 2: In the Fourier paradigm, proto-value functions are formed by diagonalizing a diffusion model. Shown here are four "smoothest" eigenvectors of the normalized graph Laplacian in a two-room MDP of 100 states.

tors. A simple example of a diffusion operator is the random walk operator $P_r = D^{-1}W$, where W is a symmetrized weight matrix, and D is a diagonal matrix whose entries are the row sums of W. Part of the random walk diffusion matrix for a simple grid world is shown in Figure 1. Interestingly, P_r is not a symmetric matrix, which might suggest that its spectral analysis would require dealing with complex numbers. Fortunately, it can be shown that P_r is closely related to a symmetric matrix called the *normalized Laplacian* $\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ (Chung 1997). Figure 2 shows the first four eigenvectors of the normalized graph Laplacian for a discrete MDP consisting of two "rooms" connected by a door. Note the eigenvectors clearly reveals the two room structure defined by the door, which acts as a bottleneck.

The similarity between value functions and proto-value functions can be remarkable, leading to a highly compact encoding (measured in terms of the number of basis functions needed to encode a value function). PVFs can be used in conjunction with a standard "black box" parameter estimation method, such as Q-learning (Watkins 1989) or least-squares policy iteration (LSPI) (Lagoudakis & Parr 2003) to find the best policy representable within the space of the chosen basis functions. Proto-value functions both reflect the large-scale geometry of a state space, as well as provide a *systematic* organization of the space of functions on a graph. Indeed, it is easy to show that PVFs are far superior to parametric architectures like radial basis functions (RBFs) at approximating nonlinear value functions even in simple MDPs (see Figure 3 for a simple example).

One hallmark of Fourier analysis is that the basis functions are localized in frequency, but not in time (or space). Hence, the eigenvectors of the graph Laplacian are localized in frequency by being associated with a specific eigenvalue λ, but their support is in general the whole graph. An alternative way to construct proto-value functions is to use *wavelets* (Mallat 1989), which are compact multiscale basis functions. Investigated mainly in Euclidean spaces, they have recently been extended to graphs and other discrete spaces. *Diffusion wavelets* provide a general way to construct multiscale proto-value functions (Mahadevan & Maggioni 2006). Diffusion wavelets are constructed not by diagonalization, but by *dilation*. Figure 4 illustrates a series of multiscale diffusion wavelet PVFs for the same two-room grid world MDP. At the bottom-most level, the diffusion wavelet basis is just the unit vector basis set (which is essentially "table-lookup"). At each succeeding level, diffusion wavelet bases are produced by using (dyadic) powers of the random walk diffusion operator (e.g. the random walk operator P_r or the normalized Laplacian L_d). Figure 4 illustrate diffusion wavelet PVFs at levels 4 and 7. At higher levels, these bases start to resemble eigenvectors, becoming progressively more global.

The remainder of the paper elaborates on the above framework, introducing a new class of algorithms that combine the learning of plan representations as well as plans, provides a deeper mathematical explanation of the ideas, and summarizes recent extensions.

Learning Representation and Behavior

This section summarizes a new class of planning algorithms called generically *Representation Policy Iteration* (RPI) (Mahadevan 2005b), because they *simultaneously* learn plan representations along with plans. Figure 5 sketches the overall algorithmic framework. There are three main components: sample collection, basis construction, and policy learning. Sample collection requires a task specification, which comprises of a domain simulator (or alternatively a physically embodied agent like a robot), and an initial policy. In the simplest case, the initial policy can be a random

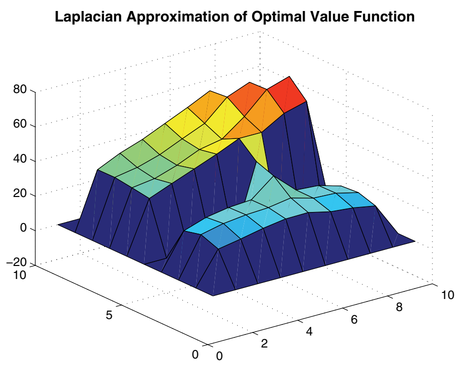


Figure 3: Approximation using 20 PVFs of the optimal value function for a two-room grid MDP of 100 states. Standard parametric bases such as polynomials or radial basis functions approximate such nonlinear value functions very poorly.

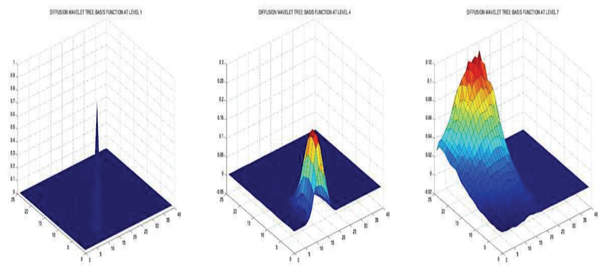


Figure 4: In the wavelet paradigm, PVFs are formed by dilating unit vectors using powers of the diffusion operator. Left: a diffusion wavelet PVF at level 1 for the two-room grid world MDP. Middle: a diffusion wavelet PVF at level 4 Right: a diffusion wavelet PVF at level 7.

walk, although it can also reflect a more informative hand-coded policy. The second phase involves constructing the bases from the collected samples using a diffusion model, such as an undirected (or directed) graph. This process involves finding the eigenvectors of a symmetrized graph operator such as the graph Laplacian. The final phase involves estimating the "best" policy representable in the span of the basis functions constructed (we are primarily restricting our attention to linear architectures, where the value function is a weighted linear combination of the bases). The entire process can then be iterated.

The results of running the algorithm on a 100 state "two-room" MDP are shown in Figure 3. This result with the following specific parameter choices.

- 4051 samples were collected using off-policy sampling from a random walk of 50 episodes, each of length 100 (or terminating early when the absorbing goal state was reached, which was the upper right hand state in Room 2). Four actions (compass direction movements) were possible from each state. Actions were stochastic. If a move-

`RPI` $(\pi_m, T, N, \epsilon, k, \mathcal{O}, \mu, \mathcal{D})$:

// π_m: Policy at the beginning of trial m
// T: Number of initial random walk trials
// N: Maximum length of each trial
// ϵ : Convergence condition for policy iteration
// k: Number of proto-value basis functions to use
// $\mathcal{O}$: Type of graph operator used
// μ: Parameter for basis adaptation
// $\mathcal{D}$: Initial set of samples

Sample Collection Phase

1. **Off-policy or on-policy sampling:** Collect a data set of samples $\mathcal{D}_m = \{(s_i, a_i, s_{i+1}, r_i), \ldots\}$ by either randomly choosing actions (off-policy) or using the supplied initial policy (on-policy) for a set of T trials, each of maximum N steps (terminating earlier if it results in an absorbing goal state), and add these transitions to the complete data set $\mathcal{D}$.
2. **(Optional) Subsampling step:** Form a subset of samples $\mathcal{D}_s \subseteq \mathcal{D}$ by some subsampling method such as random subsampling or trajectory subsampling. For episodic tasks, optionally prune the trajectories stored in $\mathcal{D}_s$ so that only those that reach the absorbing goal state are retained.

Representation Learning Phase

3. Build a diffusion model from the data in $\mathcal{D}_s$. In the simplest case of discrete MDPs, construct an undirected weighted graph G from $\mathcal{D}$ by connecting state i to state j if the pair (i, j) form temporally successive states $\in S$. Compute the operator $\mathcal{O}$ on graph G, for example the normalized Laplacian $\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$.
4. Compute the k smoothest eigenvectors of $\mathcal{O}$ on the graph G. Collect them as columns of the basis function matrix Φ, a $|S| \times k$ matrix. The state action bases $\phi(s, a)$ can be generated from rows of this matrix by duplicating the state bases $\phi(s)$ $|A|$ times, and setting all the elements of this vector to 0 except for the ones corresponding to the chosen action.[a]

Control Learning Phase

5. Using a standard parameter estimation method (e.g. Q-learning or LSPI), find an ϵ-optimal policy π that maximizes the action value function $Q^\pi = \Phi w^\pi$ within the linear span of the bases Φ using the training data in $\mathcal{D}$.
6. **Optional:** Set the initial policy π_{m+1} to π and call `RPI` $(\pi_{m+1}, T, N, \epsilon, k, \mathcal{O}, \mu, \mathcal{D})$.

[a]In large continuous and discrete MDPs, the basis matrix Φ need not be explicitly formed and the features $\phi(s, a)$ can be computed "on demand".

Figure 5: This figure shows a generic algorithm that combines the learning of plan representation from harmonic analysis of diffusion models as well as plans (policies).

ment was possible, it succeeded with probability 0.9. Otherwise, the agent remained in the same state.

- An undirected graph was constructed from the sample transitions, where the weight matrix W is simply the adjacency $(0,1)$ matrix. The normalized Laplacian $\mathcal{L} = D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}$ is then computed.
- 20 eigenvectors corresponding to the smallest eigenvalues of $\mathcal{L}$ (multiplied by 4, one set for each action) are chosen as the columns of the state action basis matrix Φ. For example, the first four eigenvectors are shown in Figure 2. Note how the eigenvectors reveal the geometric structure of the overall environment (e.g. the second eigenvector allows partitioning the two rooms since it is negative for all states in the first room, and positive for states in the second room).
- The parameter estimation method used was LSPI, with $\gamma = 0.8$.

Diffusion Analysis

We now present a deeper analysis of the graph Laplacian, specifically motivating its connection to spectral analysis of Markov decision processes. We begin with a brief overview of MDPs, and then introduce the spectral analysis of (diagonalizable) stochastic transition matrices. Diffusion models are then viewed as providing a "surrogate" model that leads to useful plan representations. It is possible to model nonsymmetric actions and policies using more sophisticated symmetrization procedures (Chung 2005), and this extension is discussed below.

Brief Overview of MDPs

A discrete Markov decision process (MDP) $M = (S, A, P^a_{ss'}, R^a_{ss'})$ is defined by a finite set of discrete states S, a finite 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). In continuous Markov decision processes, the set of states $\subseteq \mathbb{R}^d$. Abstractly, a value function is a mapping $S \rightarrow \mathbb{R}$ or equivalently (in discrete MDPs) a vector $\in \mathbb{R}^{|S|}$. 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

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

where $R_{sa} = \sum_{s' \in s} P^a_{ss'} R^a_{ss'}$ is the expected immediate reward. The expected long-term discounted sum of rewards at each state is called a value function, defined by a fixed (deterministic) policy π as

$$V^\pi(s) = R_{s\pi(s)} + \gamma \sum_{s' \in S} P^{\pi(s)}_{ss'} V^\pi(s') \qquad (1)$$

A value function in an MDP can be viewed as the result of rewards " diffusing" through the state space, governed by the underlying system dynamics. Let P^π represent an $|S| \times |S|$ transition matrix of a (deterministic) policy $\pi : S \rightarrow A$ mapping each state $s \in S$ to a desired action $a = \pi(s)$. Let R^π be a (column) vector of size $|S|$ of rewards. The value function associated with policy π satisfies the Bellman equation:

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = \left(I + \gamma P^\pi + \gamma^2 (P^\pi)^2 + \ldots \right) R^\pi \qquad (2)$$

Value functions represent the *long-term* accumulation of rewards at each state, modulated by the transition process P^π. Value functions in addition satisfy two key properties: they are typically *smooth*, [1] and they usually reflect the geometry of the environment (as illustrated in Figure 3). Smoothness derives from the fact that the value at a given state $V^\pi(s)$ is always a function of values at "neighboring" states. Consequently, it is entirely natural to construct basis functions for approximating value functions that share the same two properties. In addition, basis functions that are derived from a large-time scale analysis of an environment might be especially suitable for approximating value functions.

Let us define a set of *basis functions* $\Phi = \{\phi_1, \ldots, \phi_k\}$, where each basis function represents a "feature" $\phi_i : S \rightarrow \mathbb{R}$. The basis function matrix Φ is an $|S| \times k$ matrix, where each column is a particular basis function evaluated over the state space, and each row is the set of all possible basis functions evaluated on a particular state. Approximating a value function using the matrix Φ can be viewed as projecting the value function onto the column space spanned by the basis functions ϕ_i,

$$V^\pi \approx \Phi w^\pi = \sum_i w^\pi_i \phi_i$$

Spectral Analysis of Transition Matrices

In this paper, basis functions are constructed using a *spectral* approach, that is diagonalizing a diffusion matrix by finding its *eigenvectors*. This approach can be motivated by first assuming that the eigenvectors are constructed directly from a (known) state transition matrix P^π, and then introduce the concept of diffusion matrices that will be used instead. One subclass of diagonalizable transition matrices are those corresponding to *reversible* Markov chains. Although transition matrices for general MDPs are *not* reversible, and their spectral analysis is more delicate, it will still be a useful starting point to understand diffusion matrices such as the graph Laplacian. If the transition matrix P^π is diagonalizable, there is a complete set of eigenvectors $\Phi^\pi = (\phi^\pi_1, \ldots \phi^\pi_n)$ that provides a change of basis in which the transition matrix P^π is representable as a diagonal matrix. For the subclass of diagonalizable transition matrices represented by reversible Markov chains, the transition matrix is not only diagonalizable, but there is also an orthonormal basis. In other

[1]It is possible to quantify the notion of smooth functions on graphs using the Sobolev norm (Mahadevan & Maggioni 2006).

words, using a standard result from linear algebra, we have

$$P^\pi = \Phi^\pi \Lambda^\pi (\Phi^\pi)^T$$

where Λ^π is a diagonal matrix of *eigenvalues*. Another way to express the above property is to write the transition matrix as a sum of *projection matrices* associated with each eigenvalue:

$$P^\pi = \sum_{i=1}^{n} \lambda_i^\pi \phi_i^\pi (\phi_i^\pi)^T$$

where the eigenvectors ϕ_i^π form a complete orthogonal basis (i.e. $\| \phi_i^\pi \|_2 = 1$ and $\langle \phi_i^\pi, \phi_j^\pi \rangle = 0, i \neq j$). It readily follows that powers of P^π have the same eigenvectors, but the eigenvalues are raised to the corresponding power (i.e., $(P^\pi)^t \phi_i^\pi = (\lambda_i^\pi)^t \phi_i^\pi$). Since the basis matrix Φ spans all vectors on the state space S, we can express the reward vector R^π in terms of this basis as

$$R^\pi = \Phi^\pi \alpha^\pi$$

where α^π is a vector of scalar weights. For high powers of the transition matrix, the projection matrices corresponding to the largest eigenvalues will dominate the expansion. Combining the above equation with Equation 2, we get

$$\begin{aligned}
V^\pi &= \sum_{i=0}^{\infty} (\gamma P^\pi)^i \Phi^\pi \alpha^\pi \\
&= \sum_{k=1}^{n} \sum_{i=0}^{\infty} \gamma^i (P^\pi)^i \phi_k^\pi \alpha_k^\pi \\
&= \sum_{k=1}^{n} \sum_{i=0}^{\infty} \gamma^i (\lambda_k^\pi)^i \phi_k^\pi \alpha_k^\pi \\
&= \sum_{k=1}^{n} \frac{1}{1 - \gamma \lambda_k^\pi} \phi_k^\pi \alpha_k^\pi = \sum_{k=1}^{n} \beta_k \phi_k^\pi
\end{aligned}$$

where we used the property that $(P^\pi)^i \phi_j^\pi = (\lambda_j^\pi)^i \phi_j^\pi$. Essentially, we have now expressed the value function P^π as a linear combination of eigenvectors of the transition matrix. In order to provide the most efficient approximation, we can truncate the summation by choosing some small number $m < n$ of the eigenvectors, preferably those for whom β_k is large. Of course, since the reward function is not known, it might be difficult to pick a priori those eigenvectors that result in the largest coefficients. A simpler strategy instead is to focus on those eigenvectors for whom the coefficients $\frac{1}{1-\gamma\lambda_k^\pi}$ are the largest. In other words, the eigenvectors corresponding to the *largest* eigenvalues of the transition matrix P^π should be selected (since the spectral radius is 1, the eigenvalues closest to 1 will dominate the smaller ones).

$$V^\pi \approx \sum_{k=1}^{m} \frac{1}{1 - \gamma \lambda_k^\pi} \phi_k^\pi \alpha_k^\pi \qquad (3)$$

where the eigenvalues are ordered in non-increasing order, so λ_1^π is the largest eigenvalue . If the transition matrix P^π of a given policy π is known, one can of course construct basis functions by diagonalizing this matrix (see (Petrik 2007) for a discussion of this approach). However, dealing directly with the transition matrix P^π is problematic for several reasons. One, the transition matrix P^π cannot be assumed to be symmetric, in which case one has to deal with complex eigenvalues (and eigenvectors). Second, the transition matrix may not be known. Of course, one can always use samples of the underlying MDP generated by exploration to estimate the transition matrix, but the number of samples needed may be large. Finally, in control learning, the policy keeps changing, causing one to have to reestimate the transition matrix. Fortunately, these limitations can be overcome by using *diffusion models*, which are fairly easy to learn, are diagonalizable, and the resulting eigenvectors provide an efficient representation to approximate value functions.

From Transition Matrices to Diffusion Models

In the simplest setting, a diffusion model is just an unweighted adjacency matrix W connecting two states i and j if it is possible to reach state i from j. Note the random walk matrix $P_r = D^{-1}W$ is not symmetric. To facilitate the spectral analysis of diffusion models, it is convenient to "symmetrize" them, exploiting the property from linear algebra that symmetric matrices have real eigenvalues, and more importantly, the resulting eigenvectors are orthonormal ("perpendicular") and form a complete basis for the set of all functions on the graph G. The random walk matrix $P_r = D^{-1}W$ is called a *diffusion model* because given any function f on the underlying graph G, the powers of $P_r^t f$ determine how quickly the random walk will "mix" and converge to the long term distribution (Chung 1997). It can be shown that a random walk on an undirected graphs defines a reversible Markov chain whose stationary distribution at a given vertex is given by $P(v) = \frac{d_v}{vol(G)}$, where d_v is the degree of vertex v and the "volume" $vol(G) = \sum_{v \in g} d_v$. Since the random walk matrix P is not symmetric, it is convenient to find a symmetrized diffusion model which is closely related to it spectrally. This is essentially the graph Laplacian matrix, which is introduced next.

The Graph Laplacian

For simplicity, assume the underlying state space is represented as an undirected graph $G = (V, E, W)$, where V is the set of vertices, E is the set of edges where $(u, v) \in E$ denotes an undirected edge from vertex u to vertex v. The more general case of directed graphs is discussed below. The *combinatorial Laplacian* L is defined as the operator

$$L = D - W \ ,$$

where D is a diagonal matrix called the *valency* matrix whose entries are row sums of the weight matrix W. The normalized Laplacian is defined as

$$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

To see the connection between the normalized Laplacian and the random walk matrix $P_r = D^{-1}W$, note the following identities:

$$\begin{aligned}
\mathcal{L} &= D^{-\frac{1}{2}} L D^{-\frac{1}{2}} && (4)\\
&= I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} && (5)\\
I - \mathcal{L} &= D^{-\frac{1}{2}} W D^{-\frac{1}{2}} && (6)\\
D^{-\frac{1}{2}}(I-\mathcal{L})D^{\frac{1}{2}} &= D^{-1}W && (7)
\end{aligned}$$

Hence, the random walk operator $D^{-1}W$ is similar to $I - \mathcal{L}$, so both have the same eigenvalues, and the eigenvectors of the random walk operator are the eigenvectors of $I - \mathcal{L}$ point-wise multiplied by $D^{-\frac{1}{2}}$. One rationale for choosing the eigenvectors of the Laplacian as basis functions can now be provided. In particular, if λ_i is an eigenvalue of the random walk transition matrix P_r, then $1 - \lambda_i$ is the corresponding eigenvalue of $\mathcal{L}$. Consequently, in the expansion given by Equation 3, the eigenvectors of the normalized graph Laplacian corresponding to the *smallest* eigenvalues would be selected.

There are other reasons for choosing the low-order eigenvectors of the Laplacian as well. A fundamental property of the graph Laplacian is that projections of functions on the eigenspace of the Laplacian produce globally the smoothest approximation, which respects the underlying manifold. The Laplacian $\mathcal{L}$ also acts as a *difference* operator on a function f on a graph, that is

$$\mathcal{L}f(u) = \frac{1}{\sqrt{d_u}} \sum_{v \sim u} \left(\frac{f(u)}{\sqrt{d_u}} - \frac{f(v)}{\sqrt{d_v}} \right) w_{uv} \, . \qquad (8)$$

Extensions

Many extensions of the framework proposed in this paper are being actively explored, and these extensions are briefly summarized.

Continuous MDPs

The RPI algorithm described earlier can be straightforwardly generalized to continuous MDPs, where states are elements of $\mathcal{R}^n$. Given a set of points in $\mathcal{R}^n$ from a series of random walks, a nearest neighbor method using Euclidean distance is used to construct a diffusion model. One challenge in dealing with continuous MDPs is that PVFs are only known on sampled points, and must be extended to novel points during testing. One approach to such out-of-sample extensions is to use the *Nyström* method, which is described in (Mahadevan *et al.* 2006). Figure 6 shows results from a two-dimensional control task called the inverted pendulum, a standard benchmark task. Here, states are represented as tuples $(\theta, \dot{\theta})$, where θ is the angle of the pole, and $\dot{\theta}$ is the angular velocity. PVFs outperform published results with LSPI using RBFs (Lagoudakis & Parr 2003) by several orders of magnitude on this task.

Proto-Value Functions From Directed Graphs

The construction of PVFs can be extended to more elaborate diffusion models which capture non-directionality of actions using *directed* graphs. In particular, PVFs can be constructed by diagonalizing the directed graph Laplacian (Chung 2005), which is defined as

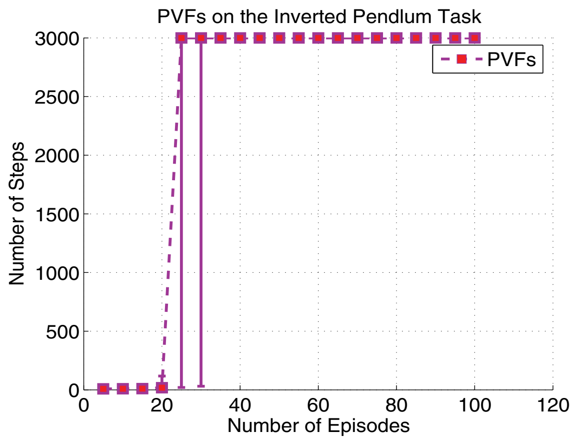


Figure 6: Performance of PVFs on the inverted pendulum continuous MDP, showing rapid learning within a few hundred steps. The plot shows median-averaged number of steps the pole was balanced over 100 learning runs. Each episode usually takes around 10 steps, and is based on executing a random policy till the pole falls over.

$$L_D = D_\phi - \frac{D_\phi P + P^T D_\phi}{2}$$

where D_ϕ is a diagonal matrix whose entries are given by $\phi(v)$, the *Perron* vector or leading eigenvector associated with the spectral radius of the transition matrix P specifying the directed random walk on G. For a strongly connected directed graph G, the Perron-Frobenius theorem can be applied to show that the transition matrix is irreducible and non-negative, and consequently the leading eigenvector associated with the largest (real) eigenvalue must have all positive components $\phi(v) > 0$. In an initial study (Johns & Mahadevan 2007), it was found that the directed graph Laplacian can result in significant improvement in some discrete and continuous MDPs. For example, in a modified two-room task where there are two "one-way" doors leading from one room to the other, PVFs constructed from the directed Laplacian significantly outperformed the non-directional PVFs constructed from undirected graphs for certain locations of the goal state (e.g. near one of the one-way doors). Directed PVFs also appeared to yield improvements in some continuous control tasks, such as the inverted pendulum.

Scaling PVFs by Kronecker Product Factorization

Proto-value functions can be made more compact using a variety of sparsification methods. One specific approach is based on Kronecker product matrix factorization (Van Loan & Pitsianis 1993). This approach has recently been implemented for continuous MDPs with promising results (Johns, Mahadevan, & Wang 2007). A random walk weight matrix $P_r = D^{-1}W$ can be approximated by a Kronecker product of two smaller stochastic matrices P_a and P_b, such that the following cost function is minimized

$$f(P_a, P_b) = \min\left(\|P_r - P_a \otimes P_b\|_2\right)$$

The approach specified in (Van Loan & Pitsianis 1993) shows how to construct two smaller stochastic matrices whose Kronecker product approximates the original random walk matrix P_r. To ensure that the decomposed matrices are not only stochastic, but also diagonalizable, which the Kronecker factorization procedure does not guarantee, an additional step using the Metropolis Hastings algorithm (Billera & Diaconis 2001) is needed to make the smaller matrices P_a and P_b *reversible*. Then, the PVFs for the original random walk matrix P_r can be approximated as the Kronecker product of the PVFs of the factorized smaller reversible matrices P_a^r and P_b^r (since the smaller matrices are reversible, they can also be symmetrized using the normalized Laplacian, which makes the numerical task of computing their eigenvectors much simpler). A preliminary study (Johns, Mahadevan, & Wang 2007), has shown that it is possible to significantly reduce the size of the random walk weight matrices for the inverted pendulum, mountain car, and the Acrobot tasks with minimal loss in performance compared to the full matrix. For example, in a four-dimensional continuous control problem called the Acrobot task, the original basis matrix is compressed by a factor of $36 : 1$, which resulted in almost as good a policy as the original larger basis matrix.

Multiscale Diffusion Wavelet Bases

One well-known limitation of global Fourier bases is that they are poor at representing piecewise linear (value) functions. Locally compact multiscale PVFs can be learned using the recently proposed *diffusion wavelet* framework (Coifman & Maggioni 2006; Bremer *et al.* 2006). Diffusion wavelets encapsulate all the traditional advantages of wavelets (Mallat 1989): basis functions have compact support, and the representation is inherently hierarchical since it is based on multi-resolution modeling of processes at different spatial and temporal scales. The performance of diffusion wavelet bases and Laplacian bases on a variety of simple MDPs is compared in (Mahadevan & Maggioni 2006). An efficient direct method for policy evaluation is presented in (Maggioni & Mahadevan 2006) which uses multiscale diffusion bases to invert the Bellman policy equation $I - \gamma P^\pi$.

Learning State-Action Basis Functions

In our paper, the basis functions $\phi(s)$ are originally defined over states, and then extended to state-action pairs $\phi(s, a)$ by duplicating the state embedding $|A|$ times and "zeroing" out elements of the state-action embedding corresponding to actions not taken. That is, $\phi(s, a) = \phi(s) \otimes I_a$ where I_a is a vector indicator function for action a (all elements of I_a are 0 except for the chosen action). This construction is somewhat wasteful, especially in domains where the number of actions can vary significantly from one state to another. PVFs can be learned instead on *state-action* graphs, where vertices represent state-action pairs. Thus, the pair (s, a) is connected by an edge to the pair (s', a') if action a in state s resulted in state s' from which action a' was next attempted. State-action graphs are naturally highly directional, and the directed Laplacian is used to compute basis functions over state action graphs. A preliminary analysis (Osentoski & Mahadevan 2007) shows that state-action bases can significantly improve the performance of PVFs in discrete MDPs.

Proto-Value Functions for Semi-Markov Decision Processes

Proto-value functions provide a broad framework for automating hierarchical reinforcement learning (Barto & Mahadevan 2003). These include the question of decomposing the overall state space by finding bottlenecks (Hengst 2002; Simsek & Barto 2004) and symmetries (Ravindran & Barto 2003). Another interesting direction is to construct PVFs for temporally extended actions, such as "exiting a room". These temporally extended actions result in "longer" edges connecting non-adjacent vertices (such as the vertices corresponding to interior states in a room with those representing the "door" state. An initial study (Osentoski & Mahadevan 2007) suggest that constructing PVFs using temporally extended actions in semi-Markov decision processes can significantly improve the performance over state-based PVFs constructed over only primitive actions.

Transfer Across Tasks

Proto-value functions are learned not from rewards, but from the topology of the underlying state space (in the "off-policy" case). Consequently, they suggest a solution to the well-known problem of transfer in reinforcement learning (Mahadevan 1992; Sherstov & Stone 2005). One key advantage of proto-value functions is that they provide a theoretically principled approach to transfer, which respects the underlying state (action) space manifold. *Proto-transfer* learning is a new framework that explores the transfer of learned representations from one task to another (in contrast to transferring learned policies) (Ferguson & Mahadevan 2006).

Policy and Reward-Sensitive Basis Functions

In the PVF framework presented above, basis functions are constructed in an *off-policy* manner without taking rewards into account. This restriction is not intrinsic to the approach, and reward or policy information when available can easily be incorporated into the construction of PVFs. One recent approach proposed in (Petrik 2007) assumes that the reward function R^π and policy transition matrix P^π are known, and combines Laplacian PVF bases with *Krlyov bases*. This approach is restricted to *policy evaluation*, which consists of solving an equation in the the well-studied form $Ax = b$, and Krylov bases are used extensively in the solution of such linear systems of equations. Another way to incorporate reward-sensitive information into PVFs is to modify the weight matrix W to take into account the *gradient* of the value function to be approximated.

Theoretical Analysis

Theoretical guarantees on the efficiency of proto-value functions in approximating value functions are being investigated. The approximation produced by projecting a given function on a graph to the smallest k proto-value functions produces *globally* the smoothest approximation (Mahadevan & Maggioni 2006). (Belkin & Niyogi 2005) show that under uniform sampling conditions, the graph Laplacian (constructed in a certain way) converges to the Laplace-Beltrami operator on the underlying manifold. Another interesting direction is to investigate the stability of the subspaces defined by proto-value functions using the tools of matrix perturbation theory (Stewart & Sun 1990).

Summary

This paper describes a unified framework for learning plan representations and plans in Markov decision processes using harmonic analysis of diffusion models. In the Fourier paradigm, proto-value functions are constructed by diagonalization of a symmetric diffusion operator on samples collecting during a random walk of the underlying state space. In the wavelet paradigm, proto-value functions are constructed by dilating the unit vector bases using powers of the diffusion operator. A general algorithmic framework called representation policy iteration (RPI) was presented consisting of three components: sample collection, basis function construction, and control learning. Several directions for scaling the approach were described.

References

Barto, A., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Systems Journal* 13:41–77.

Belkin, M., and Niyogi, P. 2005. Towards a Theoretical Foundation for Laplacian-Based Manifold Methods. In *Proceedings of the International Conference on Computational Learning Theory*.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Belmont, Massachusetts: Athena Scientific.

Billera, L., and Diaconis, P. 2001. A geometric interpretation of the Metropolis-Hasting algorithm. *Statistical Science* 16:335–339.

Bremer, J. C.; Coifman, R. R.; Maggioni, M.; and Szlam, A. D. 2006. Diffusion wavelet packets. *Applied and Computational Harmonic Analysis* 21:95–112.

Chung, F. 1997. *Spectral Graph Theory*. Number 92. CBMS-AMS.

Chung, F. 2005. Laplacians and the Cheeger Inequality for Directed Graphs. *Annals of Combinatorics*.

Coifman, R. R., and Maggioni, M. 2006. Diffusion wavelets. *Applied and Computational Harmonic Analysis* 21:53–94.

Ferguson, K., and Mahadevan, S. 2006. Proto-Transfer Learning in Markov Decision Processes using Spectral Methods. In *ICML Workshop on Transfer Learning*.

Hengst, B. 2002. Discovering hierarchy in reinforcement learning with HEXQ. In *ICML*, 243–250.

Johns, J., and Mahadevan, S. 2007. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Johns, J.; Mahadevan, S.; and Wang, C. 2007. Compact Spectral Bases for Value Function Approximation using Kronecker Factorization. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

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

Maggioni, M., and Mahadevan, S. 2006. Fast direct policy evaluation using multiscale analysis of Markov diffusion processes. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, 601–608. New York, NY, USA: ACM Press.

Mahadevan, S., and Maggioni, M. 2006. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *Proceedings of the Neural Information Processing Systems (NIPS)*. MIT Press.

Mahadevan, S.; Maggioni, M.; Ferguson, K.; and Osentoski, S. 2006. Learning Representation and Control In Continuous Markov Decision Processes. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Mahadevan, S. 1992. Enhancing transfer in reinforcement learning by building stochastic models of robot actions. In *Proceedings of the Ninth International Conference on Machine Learning, Aberdeen, Scotland*, 290–299.

Mahadevan, S. 2005a. Proto-Value Functions: Developmental Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*.

Mahadevan, S. 2005b. Representation policy iteration. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 372–37. AUAI Press.

Mahadevan, S. 2005c. Samuel Meets Amarel: Automating Value Function Approximation using Global State Space Analysis. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*. Pittsburgh: AAAI Press/MIT Press.

Mallat, S. G. 1989. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 11(7):674–693.

Osentoski, S., and Mahadevan, S. 2007. Learning state action basis functions for hierarchical MDPs. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Petrik, M. 2007. An Analysis of Laplacian Methods for Value Function Approximation in MDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

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

Ravindran, B., and Barto, A. 2003. SMDP homomorphisms: An algebraic approach to abstraction in Semi-Markov Decision Processes. In *Proceedings of the 18th IJCAI*.

Sherstov, A., and Stone, P. 2005. Improving action selection in MDP's via knowledge transfer. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*.

Simsek, Ö., and Barto, A. G. 2004. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *ICML*.

Stewart, G., and Sun, J. 1990. *Matrix Perturbation Theory*. Academic Press.

Sutton, R., and Barto, A. G. 1998. *An Introduction to Reinforcement Learning*. MIT Press.

Van Loan, C., and Pitsianis, N. 1993. Approximation with kronecker products. In *Linear Algebra for Large Scale and Real Time Applications*. Kluwer Publications. 293–314.

Watkins, C. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, King's College, Cambridge, England.