Representation Discovery in Planning using Harmonic Analysis *

Jeff Johns and **Sarah Osentoski** and **Sridhar Mahadevan**
Computer Science Department
University of Massachusetts Amherst
Amherst, Massachusetts 01003
{johns, sosentos, mahadeva}@cs.umass.edu

Abstract

This paper summarizes ongoing research on a framework for representation learning using *harmonic analysis*, a subfield of mathematics. Harmonic analysis includes Fourier analysis, where new eigenvector representations are constructed by *diagonalization* of operators, and wavelet analysis, where new representations are constructed by *dilation*. The approach is presented specifically in the context of Markov decision processes (MDPs), a widely studied model of planning under uncertainty, although the approach is applicable more broadly to other areas of AI as well. This paper describes a novel harmonic analysis framework for planning based on estimating a *diffusion model* that models flow of information on a graph (discrete state space) or a manifold (continuous state space) using a discrete form of the Laplace heat equation. 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 planning framework called Representation Policy Iteration (RPI) 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. We demonstrate the flexibility of the approach, which allows basis functions to be adapted to a particular task or reward function, and the hierarchical temporally extended nature of actions.

Motivation

The ability to learn and modify representations has long been considered a cornerstone of intelligence. The challenge of representation learning has been studied by researchers across a wide variety of subfields in AI and cognitive science, from computer vision (Marr 1982) to problem solving (Amarel 1968). In this paper, we present our ongoing research on a general framework for representation discovery that builds on recent work in *harmonic analysis*, a subfield of mathematics that includes traditional Fourier and wavelet methods (Mallat 1998). Recent work in harmonic analysis has extended the scope of these traditional analytic tools studied in Euclidean spaces to more general discrete spaces (graphs) and continuous spaces (manifolds). For example, *spectral graph theory* (Chung 1997) studies the properties of the *Laplacian*, whose eigenvectors can be viewed as a Fourier basis on graphs. Even more recently, research in computational harmonic analysis has extended multiresolution wavelet methods to graphs (Coifman & Maggioni 2006).

We build on these advances in harmonic analysis by showing how agents embedded in stochastic environments can learn novel representations for planning. We then subsequently modify or augment these representations to take into account rewards or the nature of actions. Our approach is actually more broadly applicable to a wide variety of other areas in AI, including perception, information extraction, and robotics, but we confine our presentation to planning under uncertainty. The framework provides general ways of constructing multiscale representations of stochastic processes such as Markov chains and Markov decision processes. Our approach builds on recent work in machine learning that focuses on modeling the nonlinear geometry of the space underlying many real-world datasets. Nonlinear dimensionality reduction methods have recently emerged that empirically model and recover the underlying manifold, for example multidimensional scaling (Borg & Groenen 1996), LLE (Roweis & Saul 2000), ISOMAP (Tenenbaum, de Silva, & Langford 2000), Laplacian eigenmaps (Belkin & Niyogi 2003), and diffusion maps (Coifman *et al.* 2005). These techniques can be significantly more powerful than well-studied linear Euclidean subspace methods such as principal components analysis (Jolliffe 1986).

Markov Decision Processes and Function Approximation

A Markov decision process $M = \langle S, A, P^a_{ss'}, R^a_{ss'}, \gamma \rangle$ is defined by a set of states S, a set of actions A, a transition model $P^a_{ss'}$ which specifies the probability distribution over future states s' when performing action a in state s, a reward model $R^a_{ss'}$ specifying a scalar reward, and a discount factor $\gamma \in [0, 1)$ (Puterman 1994). The discount factor balances the agent's desire for either immediate (small γ) or future

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

(large γ) rewards. A policy π is a mapping from states to actions. Given a specific policy π, a corresponding value function V^π associates a scalar value to every state in the MDP. For state s, $V^\pi(s)$ is defined as the expected long-term discounted sum of rewards received when starting in s and acting according to policy π. An optimal policy π^* is associated with a unique value function V^* that satisfies the following equation

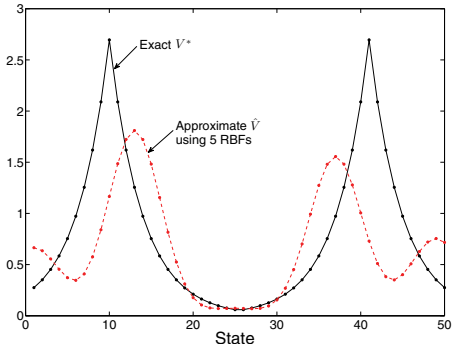
$$V^*(s) = \max_a \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma V^*(s')).$$

An exact representation of the value function stores one value for every state. For problems with large state spaces, this representation becomes infeasible. If the state space is continuous instead of discrete, then an exact representation is impossible and some form of approximation is necessary. In either situation, the challenge is to find a useful representation that scales to large problems and allows for generalization. Linear value function approximation is a common approach whereby the value function is approximated using a weighted combination of k *basis functions*

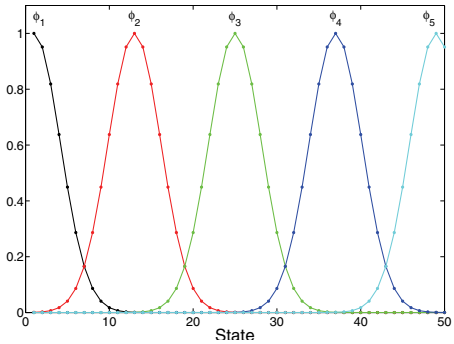
$$\hat{V}^\pi(s; \mathbf{w}) = \sum_{j=1}^{k} \phi_j(s) w_j$$

where the weights $\mathbf{w}$ are parameters. Usually $k \ll |S|$ to ensure the representation is compact. CMACs, radial basis functions (RBFs), and polynomial encodings (Sutton & Barto 1998) are often used types of basis functions. These bases can be effective but require domain specific hand engineering. Given the basis functions, there are several algorithms for learning the value function from samples of experience. Examples include temporal difference (TD) learning (Sutton 1988), least squares TD (LSTD) (Boyan & Moore 1995), and the LSTDQ algorithm in least squares policy iteration (Lagoudakis & Parr 2003). In this paper, we do not focus on the learning algorithm. The problem this paper addresses is how to automatically learn the basis functions through interaction with the domain.

To motivate our work with a concrete example, consider a simple 50-state open chain MDP. This is a MDP where an agent can transition from state s_i to s_{i+1} upon action '+' and to s_{i-1} upon action '-'. The '-' action in s_1 and the '+' action in s_{50} have no effect. The agent receives 0 reward unless it reaches states s_{10} or s_{41} where it attains a +1 reward. The optimal value function using a discount factor $\gamma = 0.8$ is shown in Figure 1(a) (solid black line). Now, consider using 5 radial basis functions (centered at states s_1, s_{13}, s_{25}, s_{37} and s_{49}) to approximate this value function. Each basis function takes the form $\phi_i(s) = \exp(\frac{(s-c_i)^2}{\sigma})$ where c_i is the center and σ is the variance of the RBF. These five basis functions are shown in Figure 1(b). The least-squares projection of the optimal value function onto these five basis functions is shown in Figure 1(a) (dashed red line). Clearly the RBF parameters could be changed to better approximate this value function, but we would like to avoid searching within this fixed parametric architecture. Rather, the goal of representation discovery in MDPs is to automatically learn the basis functions. In the rest of this paper, we present a framework for learning these representations.



(a) Exact and approximate value function



(b) Radial basis functions

Figure 1: (a) Exact value function V^* for the 50-state chain MDP (solid black) and its approximation $\hat{V}$ (dashed red) using the five radial basis functions ($\sigma = 20$) shown in (b).

Representation Learning in Markov Decision Processes

There have been several recent proposals for automatically constructing basis functions for MDPs. These methods can be categorized as either reward dependent (Keller, Mannor, & Precup 2006; Petrik 2007; Parr *et al.* 2007) or reward independent (Mahadevan 2005; Glaubius & Smart 2005).

Reward dependent approaches utlize the MDP reward function $R^a_{ss'}$ when building the basis functions. Keller et al. (2006) use the Bellman error to guide the mapping from a high dimensional state space to a lower dimensional space. The basis functions were created by aggregating states in the low dimensional space. The technique adaptively adjusts the basis subspace to represent more of the approximate value function by adding new basis functions tuned to the current Bellman error. Petrik (2007) uses the probability transition function and the reward model to create basis functions from the Krylov space vectors. Parr et al. (2007) use the Bellman error to generate basis functions. After each iteration of policy evaluation, a new basis function is added corresponding to the current estimate of the Bellman error.

In contrast, reward independent basis functions aim to capture intrinsic domain structure that should be useful for representing any smooth value function. These techniques attempt to model the manifold of the state space topology. Glabius and Smart (2005) propose using charts of a predefined size to cover the state space. Each chart's embedding function provides the bases for representing a value function. Mahadevan (2005) proposes modeling the state space topology as a graph where the connectivity is represented as a weight matrix W. W is used to form either the combinatorial Laplacian, normalized Laplacian, or the random walk operator on the graph (Chung 1997). The low order eigenvectors (i.e. the eigenvectors associated with the smallest eigenvalue) are used as basis functions. These eigenvectors are well suited to represent smooth functions because the Laplacian eigenvectors are the *smoothest* functions defined over the graph and capture nonlinearities in the domain. Use of the graph Laplacian is well justified. It is well known that the discrete graph Laplacian converges to the Laplace-Beltrami operator on Riemannian manifolds (Rosenberg 1997).

We specifically examine the Laplacian framework in this paper. While previous work using the Laplacian has been done in a reward independent manner, we show this approach is flexible and can naturally accomodate reward information.

Representation Policy Iteration

We briefly review the Representation Policy Iteration (RPI) framework (Mahadevan 2005). RPI is a novel technique for solving MDPs by simultaneously learning *both* the underlying representation and a control policy. Figure 2 is a sketch of the algorithm. This framework consists of three steps: sample collection, basis learning, and policy learning. During the sample collection phase, the agent generates a set of samples $\mathcal{D}$ by exploring the state space.

During basis learning, the samples are used to generate a graph $G = (V, E, W)$ over the state space manifold where V is the set of vertices, E is the set of edges where $(u, v) \in E$ denotes an edge from vertex u to vertex v, and W is the weighted adjacency matrix. It is important to note that the weights in W do not correspond to the transition probabilities as this would require a significant number of samples. The next step is to form a diffusion operator over the graph. This can be done using the combinatorial Laplacian, the normalized Laplacian, or the random walk operator. For an undirected graph, the combinatorial Laplacian is defined as the operator

$$L = D - W \tag{1}$$

where D is a diagonal matrix whose entries are simply equal to the row sums of the weight matrix W (e.g. $D_{ii} = \sum_j W_{ij}$). The normalized graph Laplacian, which standardizes the combinatorial Laplacian to have unit trace, is defined as

$$\mathcal{L} = D^{-1/2} L D^{-1/2}. \tag{2}$$

The random walk operator is simply defined as

$$P = D^{-1} W. \tag{3}$$

The basis functions are the *smoothest* eigenvectors of the diffusion operator. Since W is a symmetric matrix for an undirected graph, these three operators have real-valued eigenvalues and eigenvectors. Moreover, the eigenvectors form a complete basis. These basis functions are orthogonal for the two Laplacian operators as long as the underlying graph is undirected. In the case of the random walk operator, the basis functions are orthogonal with respect to the measure induced by D.

The previous description deals primarily with discrete MDPs. A few modifications are required to extend this to the case of continuous MDPs. The first issue that must be dealt with is how to construct the graph based on a set of samples in $\mathcal{R}^n$. This can be solved by connecting samples to either their k nearest neighbors or to all neighbors within a specified distance. The second issue is how to extend the basis functions to states that are not in the graph. This can be done using Nyström interpolation or nearest neighbor interpolation. A more detailed description can be found in Mahadevan et al. (2006).

Work has also been done to investigate different methods of representing the underlying manifold. Johns and Mahadevan (2007) demonstrate the effectiveness of creating basis functions from the *directed* graph Laplacian. Osentoski and Mahadevan (2007) also use the directed graph Laplacian to create basis functions in state-action space instead of state space. When building the basis functions over states, the state-action embeddings were created by copying the functions for every action. This copying is unnecessary when creating graphs directly over state-action space. Both techniques demonstrate performance improvements in some control tasks.

Basis functions of the graph Laplacian have global support. As such, these bases can have difficulty representing functions with discontinuities. An alternative method for constructing basis functions over a graph is to use diffusion wavelets (Coifman & Maggioni 2006) which are compact, multiscale bases. Diffusion wavelets are a generalization of traditional wavelets to operate on graphs. As opposed to eigenvectors which are formed by diagonalizing a diffusion operator, diffusion wavelets are computed by dilation. At the finest resolution, the diffusion wavelet basis is simply the unit vector basis on the graph. More coarse bases are generated by taking powers of the diffusion operator. The important property that makes the coarse bases more compact (than the unit vector basis) is that higher order powers of the diffusion operator are increasingly rank-deficient. We refer the reader to (Coifman & Maggioni 2006) and (Mahadevan & Maggioni 2006) for a detailed description of the diffusion wavelet construction.

The third step in the RPI framework is the *control learning* step. The agent uses a learning algorithm such as least squares policy iteration (LSPI) (Lagoudakis & Parr 2003) or Q-learning to learn the best policy representable within the space defined by the basis functions.

To demonstrate the representations, we present the first four non-constant Laplacian eigenvectors in four domains: a simple 50-state open chain (Figure 3), a 20x20 grid domain (Figure 4), and the mountain car domain (Figure 5).

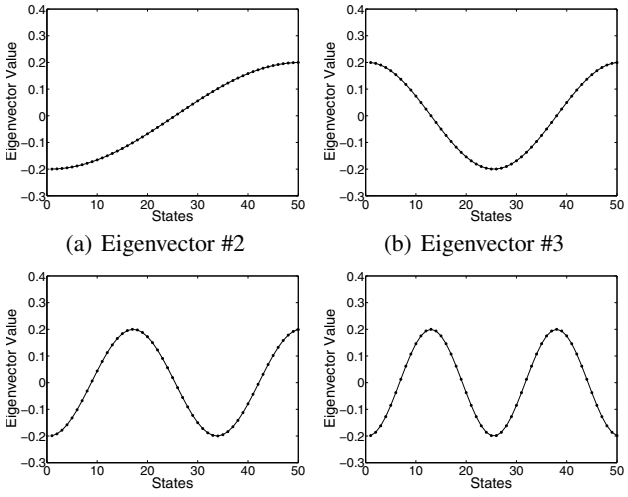
RPI Algorithm $(\mathcal{D}, \gamma, \epsilon, k, \pi_0)$:

1. **Sample Collection:**
   (a) Generate a set of samples $\mathcal{D}$ which consists of a state, action, reward, and nextstate, (s, a, r, s'). The samples are created using a series of random walks in the environment. The random walks terminate when an absorbing state is reached or some preset maximum number of steps is reached.
   (b) Subsample $\mathcal{D}$ if necessary in order to gain a smaller set of transitions $\mathcal{D}_s$.
2. **Representation Learning:**
   (a) Build an undirected weighted graph G from $\mathcal{D}_s$ where V is the set of vertices, E edge set, and W is the weight matrix. This can be done by connecting vertices that are temporally linked in $\mathcal{D}_s$ with an edge of weight 1. Another method to connect states is to use a k nearest neighbor algorithm where edge weights can be set using $W(i,j) = W(j,i) = \nu(i)e^{-\frac{d(s_i,s_j)^2}{\sigma}}$ where $\sigma > 0$ is a parameter, ν is a weight function that must be specified, and $d(s_i, s_j)$ is an appropriate distance metric.
   (b) Use the graph to generate the diffusion operator via Equation 1, 2, or 3.
   (c) Calculate the k smoothest eigenfunctions of the diffusion operator. These k eigenvectors form the basis functions $[\phi_1, \phi_2, \ldots, \phi_k]$.
3. **Control Learning**:
   Use a parameter estimation method such as LSPI (Lagoudakis & Parr 2003) or Q-learning (Watkins 1989) to find the best policy π.

   Initialize $w^0 \in \mathcal{R}^k$ to a random vector. $\pi' = \pi_0, w = w_0$
   Repeat the following steps **until** $\|w^i - w^{i+1}\|^2 < \epsilon$.
   (a) $\pi_t = \pi'$
   (b) $\pi' = \text{LSPI}(D, k, \phi, \gamma, \pi)$
   (c) $t = t + 1$

Figure 2: RPI Framework for learning representation and control.

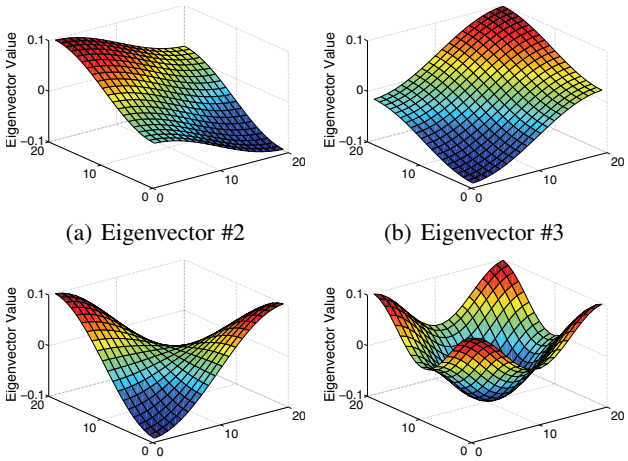
The mountain car domain is a classic benchmark MDP with a two dimensional continuous state space (position and velocity). The goal of the mountain car task is to get a simulated car to the top of a hill as quickly as possible (Sutton & Barto 1998). The car does not have enough power to get there immediately so it must oscillate on the hill to build up the necessary momentum. We also present two diffusion wavelets for each of these domains in Figures 6, 7, and 8. One of the wavelets is a more localized function and the other is more global in nature. The global diffusion wavelets demonstrate that this technique does in fact learn the lowest frequency Laplacian eigenfunctions at the most coarse level of the wavelet tree.



(a) Eigenvector #2 (b) Eigenvector #3

(c) Eigenvector #4 (d) Eigenvector #5

Figure 3: Smoothest eigenvectors of 50-state open chain.



(a) Eigenvector #2 (b) Eigenvector #3

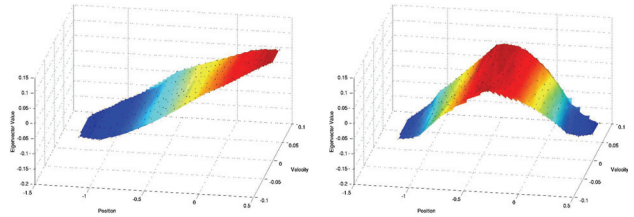
(c) Eigenvector #4 (d) Eigenvector #5

Figure 4: Smoothest eigenvectors of 20x20 grid.

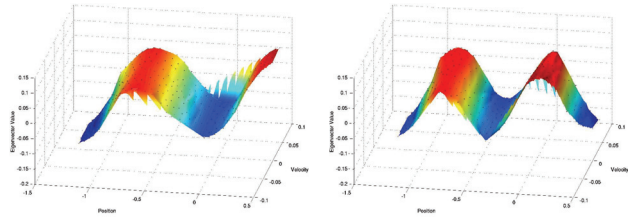
Adapting Basis Functions to Exploration

In continuous state MDPs, it is often the case that an agent experiences new states which it has never seen (exactly) before. Clearly it would be very inefficient if the agent had to add every new state to the graph and recompute the basis functions. This degree of resolution is unnecessary. If the new state is sufficiently close to states contained in the graph, then the representation for that state can be computed using the Nyström interpolation method or other schemes.

The Nyström method interpolates the value of eigenvectors computed on sample states to novel states and is an application of a classical method used in the numerical solution of integral equations (Baker 1977). It can be viewed as a technique for approximating a semi-positive definite matrix from a low-rank approximation. The basic idea is to approximate each eigenvector at the new point based on a weighted sum of the eigenvectors associated with nearby states in the

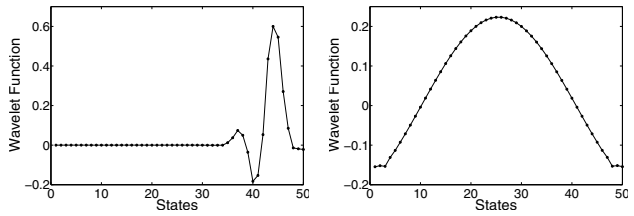


(a) Eigenvector #2 (b) Eigenvector #3



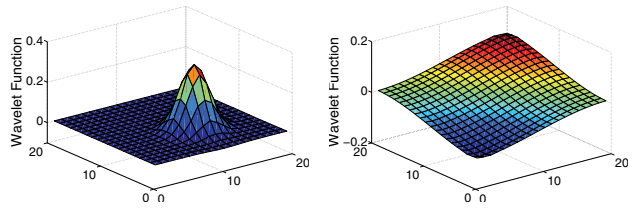
(c) Eigenvector #4 (d) Eigenvector #5

Figure 5: Smoothest eigenvectors in mountain car.



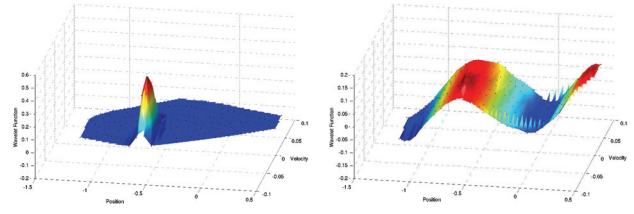
(a) Localized function (b) Global function

Figure 6: Examples of diffusion wavelet bases in 50-state open chain.



(a) Localized function (b) Global function

Figure 7: Examples of diffusion wavelet bases in 20x20 grid.



(a) Localized function (b) Global function

Figure 8: Examples of diffusion wavelet bases in mountain car.

graph. Further details on the Nyström method can be found in (Mahadevan *et al.* 2006) and (Drineas & Mahoney 2005).

If new states are not well represented by states contained in the graph, then it becomes necessary to explicitly grow the graph. This can be done quickly. Assuming the Nyström method has been run, the approximate *extended* eigenvectors (e.g. the original eigenvectors with approximate values for the new states in the graph) can be used to initialize an iterative eigensolver. This ensures previous results are not completely lost. Within a small number of iterations, the eigensolver will refine these initial approximate eigenvectors into more precise eigenvectors on the larger graph. The extra cost of this computation is $O(IN)$ if I iterations are necessary and if the adjacency matrix of the extended graph is sparse (e.g. only $O(N)$ non-zero entries).

Adapting Basis Functions to Reward Information

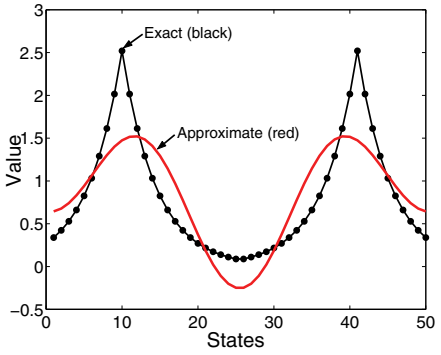
So far, the learned basis functions have been derived purely from the topology of the graph and irrespective of the MDP's reward structure. For the graph Laplacian, this property ensures the eigenvectors associated with smaller eigenvalues are smoother than eigenvectors associated with larger eigenvalues. However, it may be advantageous to allow the representation to account for the reward function. This is the central idea behind several basis function construction algorithms that incrementally add basis functions based on the Bellman error (Menache, Mannor, & Shimkin 2005; Keller, Mannor, & Precup 2006; Parr *et al.* 2007). In terms of the Laplacian eigenfunctions, there are several reasons why accounting for the reward function could improve the representation. First, the smoothness of the value function may vary across regions of the state space (e.g. an action in one region of the state space may have a larger impact on the state than in a different region). Second, for certain domains, the value function may contain discontinuities. For example, it is well known that in the mountain car domain (Sutton & Barto 1998) there is a steep ridge in the value function (corresponding to states that can just barely reach the top of the hill versus states that must go back down the hill to build up more momentum). One of the advantages that the graph data structure offers is the ability to seamlessly include this type of information once it becomes known (i.e. to change the representation based on the function being learned). We propose accounting for the reward function by adjusting the graph's edge weights, which were originally set based on topological distance, to include another term accounting for smoothness in the value function. This is formalized by adjusting the edge weights as follows

$$W(i,j) = \exp\left(-\frac{d(\mathbf{s}_i,\mathbf{s}_j)^2}{\sigma_1} - \frac{f(V_{s_i},V_{s_j})}{\sigma_2}\right) \qquad (4)$$

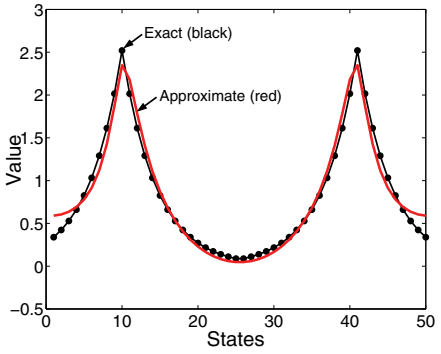
where $d(\mathbf{s}_i,\mathbf{s}_j)^2$ is the usual distance function between two states and $f(V_{s_i},V_{s_j})$ is a nonnegative function based on the shape of the value function at vertices $\mathbf{s}_i$ and $\mathbf{s}_j$. We use $f(V_{s_i},V_{s_j}) = |V_{s_i} - V_{s_j}|$ for the plots in this section. The parameters σ_1 and σ_2 can be set according to the scale of the data and function space. In fact, these parameters can be adjusted over time as the value function becomes more well known. This idea was recently introduced in the machine

learning literature as a way to produce "data and function dependent kernels" (Szlam, Maggioni, & Coifman 2006). It is a way to focus the representational power of the function approximator in the locations that need it most.

To demonstrate the potential effectiveness of this change of representation, we present an example using the same 50-state open chain MDP used in this paper's introduction. Figure 9(a) shows the exact value function and its approximation using the first 5 basis functions derived from the graph Laplacian. The edge weights were set to $\sigma_1 = 1.0$ and $\sigma_2 = \infty$. Five basis functions were again used for the approximation in Figure 9(b), but this time the edge weights were set to $\sigma_1 = 1.0$ and $\sigma_2 = 0.15$. In this example, the basis subspace is clearly improved by incorporating both the data and the function in the edge weights.



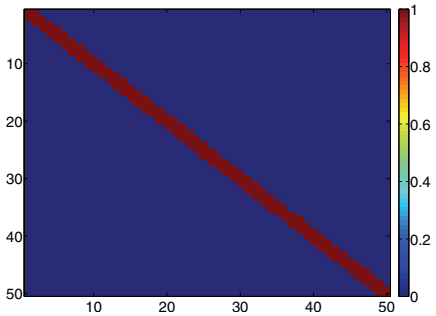
(a) Approximation with data dependent basis functions



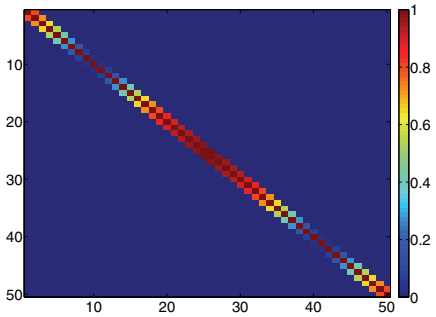
(b) Approximation with data and function dependent basis functions

Figure 9: Exact (black) versus approximate (red) value function using 5 basis functions.

To understand these results, it is instructive to contrast the data dependent basis functions with the data and function dependent bases. Figure 10 shows the data dependent weight matrix (e.g. all edges have a weight of 1) with the data and function dependent weight matrix. Notice the edges near the goal states have smaller weights. This impact on the second through fifth eigenvectors of the 50-state chain MDP is shown in Figure 11. Again, the first eigenvector is omitted because it is simply a constant. When accounting for both the data and the value function, the basis functions change more quickly near the two goal states. This occurs because the value function is changing more quickly in magnitude near the two goal states.

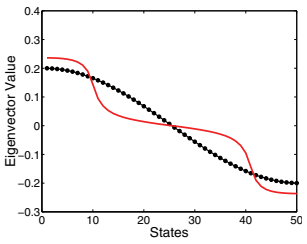


(a) Task independent W

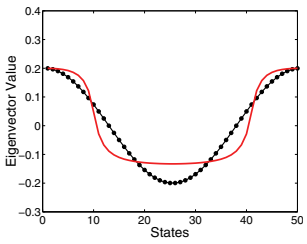


(b) Reward sensitive W

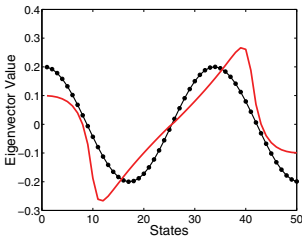
Figure 10: (a) Canonical weight matrix of all ones for the 50-state chain MDP, (b) adapted weight matrix using Equation 4 with $\sigma_1 = \infty$ and $\sigma_2 = 0.15$.



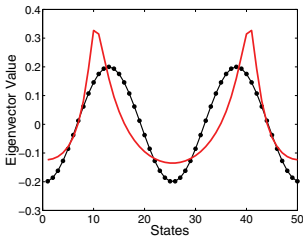
(a) Eigenvector #2



(b) Eigenvector #3



(c) Eigenvector #4



(d) Eigenvector #5

Figure 11: Smoothest eigenvectors of 50-state open chain. The two curves are when the graph's edge weights are set based on the data (black) versus being set based on the data and the function (red).

This simple example shows there may be benefits to adapting the basis functions to a specific function. However, the graph's edge weights were adjusted in Equation 4 by using the exact value function which we clearly do not have access to in practice. The most obvious way to implement this basis adaptation idea within a reinforcement learning algorithm is to initially set $\sigma_2 = \infty$. Then, once the algorithm computes an estimated value function, recompute the edge weights with an appropriate $\sigma_2 < \infty$. The basis functions then have to be recomputed which can be expensive, but there may be ways to reuse previous intermediate computation. The learning algorithm can then be rerun with the new basis functions. Although this extension to include reward information in the graph is fairly straightforward, our previous work was done independent of the reward. We believe the flexibility to be either reward dependent or independent is a strength of the graph based framework.

A significant challenge for this paradigm is how to select the function $f(\hat{V}_{s_i}, \hat{V}_{s_j})$ in Equation 4. Clearly, the approximate gradient $|\hat{V}_{s_i} - \hat{V}_{s_j}|$ may not coincide with the exact gradient $|V^*_{s_i} - V^*_{s_j}|$ depending on the current basis function subspace. In fact the gradients can be far apart. We are currently exploring whether there is a practical, principled way to adjust the weights to better represent the value function.

Basis Functions for Hierarchical Abstractions

The previous two sections discussed adjustments to the basis functions that worked for both the Laplacian eigenvectors and the diffusion wavelets. This section discusses an idea unique to diffusion wavelets. As mentioned in the introduction, diffusion wavelets are a multilevel representation that capture both spatial and temporal abstractions. This localization of basis functions can result in improved function approximation performance compared to the global Laplacian eigenfunctions because discontinuities can be isolated instead of having a global effect on the approximation.

Aside from the function approximation benefits of localized bases, diffusion wavelets can also be thought of purely as a hierarchical representation of a Markov process. There are many possible ways this data structure can be employed. For example, Maggioni and Mahadevan (2006) demonstrated that diffusion wavelets, which efficiently represent powers of the transition matrix P, can be used to quickly invert the Green's function $(I - \gamma P)$. This is an important step in evaluating a policy. There is also an interesting connection between diffusion wavelets and the options framework (Sutton, Precup, & Singh 1999). Options are temporally extended actions that significantly speed up learning and planning by searching at a higher level of abstraction. An option consists of three components: an initiation set which contains the states from which the option can be invoked, a policy which determines action selection, and a termination condition which specifies the probability of finishing the option in any state. Diffusion wavelets can be used to automatically generate options.

There have been several proposed algorithms for identifying subgoals in order to automatically generate options in reinforcement learning tasks (Şimşek, Wolfe, & Barto 2005; McGovern & Barto 2001; Menache, Mannor, & Shimkin 2002). For example, Şimşek and Barto (2005) proposed finding subgoals by partitioning *local transition graphs*. The rationale for doing so is that useful partitions lead to the identification of *access states* (e.g. states that allow the agent to move between partitions) which in turn can be used to (1) more efficiently explore the state space and (2) create options that can be used for multiple tasks. It is relatively straightforward to apply diffusion wavelets to subgoal identification. Diffusion wavelets are constructed by taking powers of the graph's diffusion operator. At higher powers, the diffusion operator in effect clusters states. Access states thus naturally appear in the multiple time scale hierarchy when clusters eventually merge. An interesting observation is that diffusion wavelets seem to provide a principled way to form initiation sets for options as compared to the local graph partitioning framework (Şimşek, Wolfe, & Barto 2005) where the initiation set depends on the length of the state transition history used to generate the local graph (which is a manually chosen parameter). Another benefit of applying the diffusion wavelet framework to options is that once an option is generated, the wavelets naturally provide a set of basis functions for the option. These basis functions can be used for learning the option policy and for compactly representing an option model.

Conclusions and Future Work

In this paper we presented a framework for discovering representations that are useful for solving MDPs. We showed that basis functions can be automatically generated from diffusion operators defined over MDPs. Eigenvectors of the diffusion operator generate global functions via diagonalization whereas diffusion wavelets generate multiscale basis functions via dilation. These bases are smooth functions which are useful for representing smooth value functions. This is an important property of a representation for MDPs because it can be used within an approximate policy iteration algorithm to learn control policies.

We proposed that these representations can be adapted based on the agent's changing experience. The key to this approach, whether using Laplacian eigenfunctions or diffusion wavelets, is mainly in the construction of the weighted graph which captures state space regularities. We demonstrated how the graph can be altered to encode alternate information such as the reward. It is also natural to adjust the graph for new samples that are not easily explained by the current representation. We have focused in this paper on automatic representation construction and adaptation for value function approximation. Experimental results for applying these basis functions within a reinforcement learning algorithm can be found in several of our other papers (Mahadevan *et al.* 2006; Johns & Mahadevan 2007; Osentoski & Mahadevan 2007).

There are many interesting avenues for future work. While it is evident that the graph can adapt to the agent's changing experience, this effect is not well understood. More experimental and theoretical analysis similar to that of Parr et al. (2007) would provide further insight to basis construction. In this paper we characterized the policy

dependent and policy independent techniques as opposing methodologies. However, this is not necessary and an agent that could utilize both and adapt to new reward functions without needing to relearn its entire representation is an interesting area for future research.

References

Amarel, S. 1968. On representations of problems of reasoning about actions. In Michie, D., ed., *Machine Intelligence 3*, volume 3, 131–171. Elsevier/North-Holland.

Baker, C. 1977. *The Numerical Treatment of Integral Equations*. Oxford: Clarendon Press.

Belkin, M., and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 6(15):1373–1396.

Borg, I., and Groenen, P. 1996. *Modern Multidimensional Scaling : Theory and Applications*. Springer.

Boyan, J., and Moore, A. 1995. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*. Cambridge, MA: MIT Press. 369–376.

Chung, F. 1997. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. Providence, RI: American Mathematical Society.

Coifman, R. R., and Maggioni, M. 2006. Diffusion wavelets. *Applied and Computational Harmonic Analysis* 21(1):53–94.

Coifman, R. R.; Lafon, S.; Lee, A.; Maggioni, M.; Nadler, B.; Warner, F.; and Zucker, S. 2005. Geometric diffusions as a tool for harmonic analysis and structure definition of data. Part I: Diffusion maps. *Proc. of Nat. Acad. Sci.* (102):7426–7431.

Drineas, P., and Mahoney, M. 2005. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research* 6:2153–2175.

Glaubius, R., and Smart, W. 2005. Manifold representations for value-function approximation in reinforcement learning. Technical Report 05-19, Department of Computer Science and Engineering, Washington University in St. Louis.

Johns, J., and Mahadevan, S. 2007. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the 24th International Conference on Machine Learning*. New York, NY: ACM Press.

Jolliffe, T. 1986. *Principal Components Analysis*. Springer-Verlag.

Keller, P.; Mannor, S.; and Precup, D. 2006. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, 449–456. New York, NY: ACM Press.

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 *Proceedings of the 23rd International Conference on Machine Learning*, 601–608. New York, NY: ACM Press.

Mahadevan, S., and Maggioni, M. 2006. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In Weiss, Y.; Schölkopf, B.; and Platt, J., eds., *Advances in Neural Information Processing Systems 18*, 843–850. Cambridge, MA: MIT Press.

Mahadevan, S.; Maggioni, M.; Ferguson, K.; and Osentoski, S. 2006. Learning representation and control in continuous Markov decision processes. In *Proc. of the 21st National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Mahadevan, S. 2005. Representation Policy Iteration. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 372–379. Arlington, VA: AUAI Press.

Mallat, S. 1998. *A wavelet tour in signal processing*. Academic Press.

Marr, D. 1982. *Vision: A Computational Investigation into the Human Representation and processing of Visual Information*. San Francisco, CA: Freemann.

McGovern, A., and Barto, A. G. 2001. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning*, 361–368. Morgan Kaufmann.

Menache, I.; Mannor, S.; and Shimkin, N. 2002. Q-cut - dynamic discovery of sub-goals in reinforcement learning. In *Proceedings of the 13th European Conference on Machine Learning*, 295–306. Springer-Verlag.

Menache, I.; Mannor, S.; and Shimkin, N. 2005. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operation Research* 134(1):215–238.

Osentoski, S., and Mahadevan, S. 2007. Learning state-action basis functions for hierarchical MDPs. In *Proceedings of the 24th International Conference on Machine Learning*. New York, NY: ACM Press.

Parr, R.; Painter-Wakefield, C.; Li, L.; and Littman, M. 2007. Analyzing feature generation for value-function approximation. In *Proceedings of the 24th International Conference on Machine Learning (to appear)*. New York, NY: ACM Press.

Petrik, M. 2007. An analysis of Laplacian methods for value function approximation in MDPs. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, 2574–2579.

Puterman, M. L. 1994. *Markov Decision Processes*. New York, NY: John Wiley and Sons.

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

Roweis, S., and Saul, L. 2000. Nonlinear dimensionality reduction by local linear embedding. *Science* 290:2323–2326.

Şimşek, Ö.; Wolfe, A. P.; and Barto, A. G. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*. New York, NY: ACM Press.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning*. Cambridge, MA: MIT Press.

Sutton, R.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112:181–211.

Sutton, R. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3:9–44.

Szlam, A.; Maggioni, M.; and Coifman, R. 2006. A general framework for adaptive regularization based on diffusion processes on graphs.

Tenenbaum, J.; de Silva, V.; and Langford, J. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290:2319–2323.

Watkins, C. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, University of Cambridge.