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 such as graphs and continuous spaces such as 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, and then subsequently modify or augment these representations to take into account rewards or the nature of actions. Our approach is actually much 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.

More generally, we believe this approach has fundamental and far-reaching implications for many areas of artificial intelligence. Rather than searching in a fixed representation space, it yields techniques that dynamically construct new representation spaces that greatly simplify the optimization problem of finding satisficing or optimal solutions. The framework provides general ways of constructing multiscale representations of stochastic processes such as Markov chains and Markov decision processes, clustering data on graphs and manifolds hierarchically, forming categories in high-dimensional spaces with sparsely labeled training instances, and exploiting the availability of unlabeled examples. Our approach also 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).

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

Representation Learning in Markov Decision Processes

Recent work on Representation Policy Iteration (RPI) (Mahadevan 2005) has examined a novel framework for solving Markov decision processes (MDPs) by simultaneously learning both the underlying representation for representation and the (approximate) optimal policies. In this paper we specifically examine how this technique can accommodate change in the agents policy or environment. We briefly review MDPs, and then discuss basis function construction and how the bases can be adapted to (1) accommodate new information, and (2) plan at different levels of abstraction. Lastly, we wrap up with some conclusions and plans for future work.

A MDP is defined as $M = (S, A, P^a_{ss'}, R^a_{ss'})$ where S is the set of states, A is the set of actions, and $P^a_{ss'}$ is the one-step transition probability and $R^a_{ss'}$ is the expected reward for transitioning from state s to s' under action a. A policy π is a mapping from states to actions. The optimal action value function $Q^*(s, a)$ satisfies the Bellman equation:

$$Q^*(s,a) = \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma \max_{a'} Q^*(s', a')).$$

An exact representation stores one value for each state-action pair. This table lookup approach scales exponentially with the dimensionality of the state space. If the state space is continuous, then an exact representation is impossible and some form of approximation is necessary. In either of these situations (large discrete state space or continuous state space), the challenge is to find a useful representation that scales to large problems and allows for generalization. The most typical approach is to approximate the value function using a set of k *basis functions* ($k \ll |S| \times |A|$) $\phi(s, a)$ with a linear architecture

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

where the weights $\mathbf{w}$ are tuned to minimize an error metric (often the L_2 error). The set of basis functions $\Phi = [\phi_1, \phi_2, \ldots, \phi_k]$ are defined over all possible state-action pairs that could be generated in the MDP.

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 (Parr *et al.* 2007). In this paper, we do not focus on the learning algorithm. Instead, we emphasize how to automatically learn and adjust the basis functions.

Basis Functions from Diffusion Models

Many techniques for value function approximation have been studied. The most common technique is linear function approximation where the value functions are represented as a linear combination of basis functions. CMACs, radial basis functions (RBFs), and polynomial encodings (Sutton & Barto 1998) are basis functions that are frequently used. These basis functions can be effective but require domain specific hand engineering. Recently several techniques have been proposed to automatically construct basis functions. These methods can be categorized as either policy dependent (Keller, Mannor, & Precup 2006; Petrik 2007; Parr *et al.* 2007) or policy independent (Smart 2004; Mahadevan 2005).

The policy dependent approaches all incorporate reward into the building of the basis functions. Keller et al. (Keller, Mannor, & Precup 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 that are tuned to the current Bellman error. Petrik (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 create 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, policy independent basis functions aim to capture intrinsic domain structure that should be useful for representing smooth value functions. Both Smart and Mahadevan propose techniques to model the manifold of the state space topology. Smart (2004) proposes using charts of a predefined size were allocated to cover the state space. Each chart's embedding function provided 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 or normalized graph Laplacian (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 also capture nonlinearities in the domain.

In this paper we will specifically examine the Laplacian framework. While this framework traditionally has been policy independent, we will show that the approach is flexible enough to adapt the representation as the agent's policy changes.

Representation Policy Iteration

First we will briefly review the Representation Policy Iteration (RPI) framework (Mahadevan 2005). Figure 1 sketches the overall algorithm. This framework consists of three steps: sample collection, basis learning, and policy learning. During the sample collection the agent generates a set of samples $\mathcal{D}$ by exploring the state space, typically using a random walk.

During basis learning the agent first creates a graph $G = (V, E, W)$ over the state or state-action 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 probabil-

ities as this would require a significant number of samples. The basis functions are the k smoothest eigenvectors of the undirected or directed graph Laplacian. For an undirected graph the combinatorial graph Laplacian is defined as the operator

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

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

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

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 build the graph on the set of points that are an element of $\mathcal{R}^n$ continuous space. A graph can be built by subsampling $\mathcal{D}$ and then connecting subsamples by using a nearest neighbor algorithm. 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. However, an alternative method of constructing basis functions over a graph is diffusion wavelets (Coifman & Maggioni 2006) which provide a general way to construct multiscale basis functions. Diffusion wavelets are constructed by a dilation process. Basis functions are produced by taking the powers of the random walk operator, $P_r = D^{-1}W$. At the most fine resolution, the basis is simply the unit vector basis set. At the coarsest level, the basis has global support and corresponds to the lowest frequency Laplacian eigenfunctions.

The third step is the *control learning* step. In this 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 of the chosen 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 2), a 20x20 grid domain (Figure 3), and the mountain car domain (Figure 4). 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 5, 6, and 7. 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

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

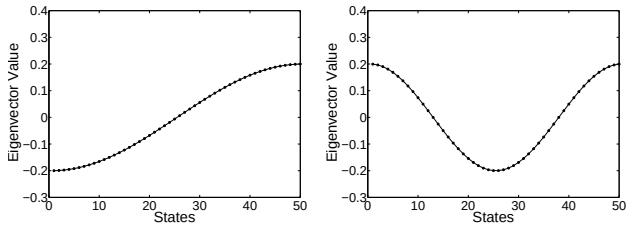
1. **Sample Collection:**
   (a) Generate a set of samples $\mathcal{D}$ which consists of a state, action, reward, and nextstate, (s, a, r, s'). The samples are created using a series of random walks in the environment. The random walks terminate when an absorbing state is reached or some preset maximum number of steps is reached.
   (b) Subsample $\mathcal{D}$ in order to gain a smaller set of transitions $\mathcal{D}_f$ by some method, random or greedy are typical examples.
2. **Representation Learning:**
   (a) Build an undirected weighted graph G from $\mathcal{D}_f$ where V is the set of vertices, E edge set, and W is the weight matrix. The vertices are the set of states, S. Several methods can be used to connect the states. The simplest technique is placing an edge with weight 1 between state i a state j if they are temporally linked in $\mathcal{D}_f$.

   Another method connects state i to state j if it is one of it's k "nearest" neighbors. The weight $w(i,j) = w(j,i) = \nu(i) e^{\frac{d(s_i, s_j)}{\sigma}}$ where $\sigma > 0$ is a parameter and ν is a weight function that must be specified. $d(s_i, s_j)$ is a distance metric such as the Euclidean or Manhattan distance.
   (b) Calculate the k lowest order eigenfunctions of the (combinatorial or normalized) graph Laplacian operator on G. These k eigenvectors make up the basis functions ϕ.
      i. Form the directed Laplacian per Equation 1 or 2.
      ii. Calculate ϕ by computing the eigenvectors of the directed Laplacian.
      Create the basis functions for state action pairs by concatenating the state encoding $|A|$ times.
3. **Control Learning Phase**:
   Use a parameter estimation method such as LSPI (Lagoudakis & Parr 2003) or Q-learning (Watkins 1989) to find the best policy π. Previous papers have primarily focused on the use of LSPI:

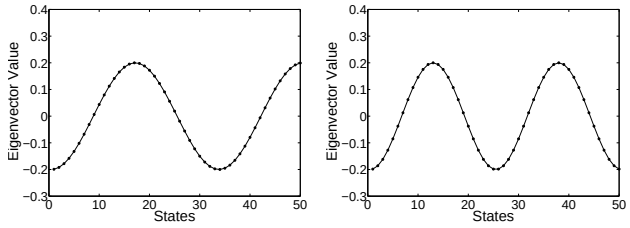
   Initialize $w^0 \in \mathcal{R}^k$ to a random vector. $\pi' = \pi_0, w = w_0$
   Repeat the following steps **until** $\|w^i - w^{i+1}\|^2 < \epsilon$.
   (a) $\pi_t = \pi'$
   (b) $\pi' = \text{LSTDQ}(D, k, \phi, \gamma, \pi)$
   (c) $t = t + 1$

Figure 1: RPI Framework for learning representation and control.

frequency Laplacian eigenfunctions at the most coarse level of the wavelet tree.

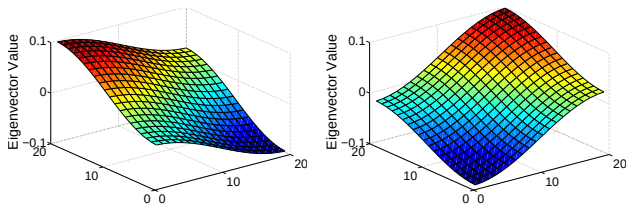


(a) Eigenvector #2 (b) Eigenvector #3

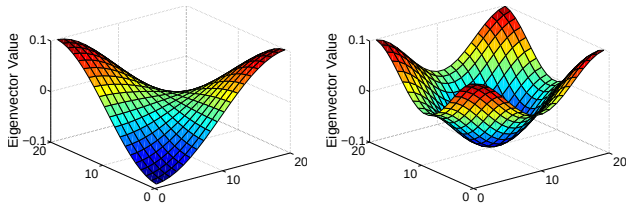


(c) Eigenvector #4 (d) Eigenvector #5

Figure 2: Smoothest eigenvectors of 50-state open chain.



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



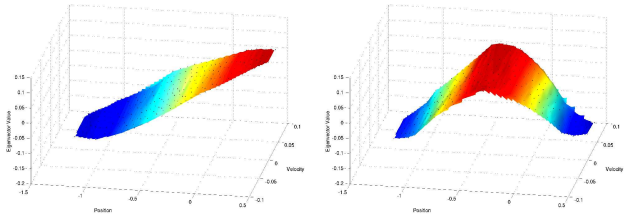
(c) Eigenvector #4 (d) Eigenvector #5

Figure 3: Smoothest eigenvectors of 20x20 grid.

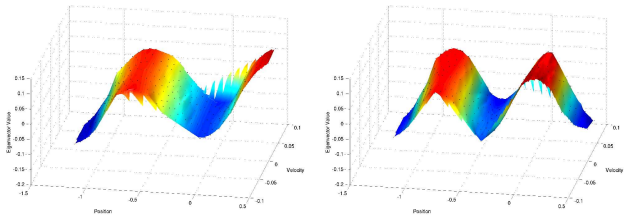
Adapting Basis Functions to Exploration

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

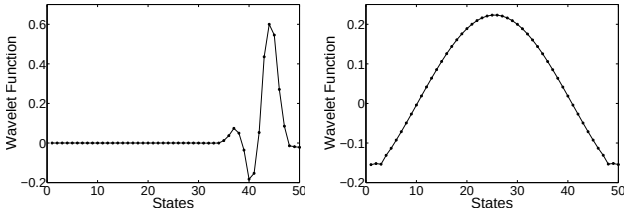


(a) Eigenvector #2 (b) Eigenvector #3



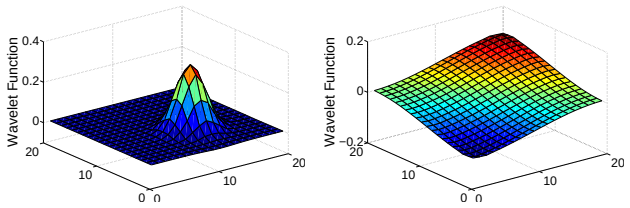
(c) Eigenvector #4 (d) Eigenvector #5

Figure 4: Smoothest eigenvectors in mountain car.



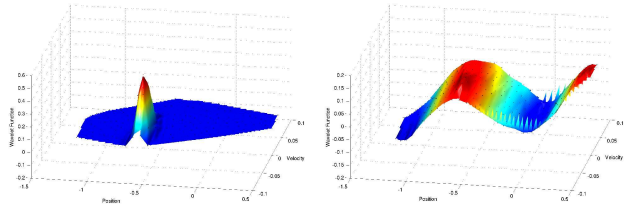
(a) Localized Function (b) Global Function

Figure 5: Examples of diffusion wavelet bases in 50-state open chain.



(a) Localized Function (b) Global Function

Figure 6: Examples of diffusion wavelet bases in 20x20 grid.



(a) Localized Function (b) Global Function

Figure 7: Examples of diffusion wavelet bases in mountain car.

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 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

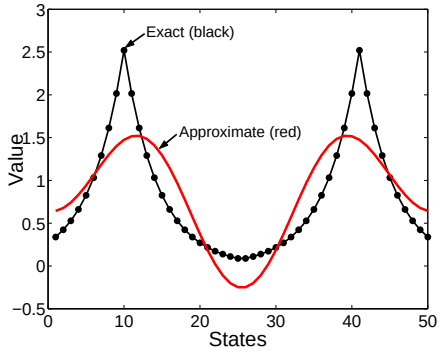
So far, the learned basis functions have been derived purely from the topology of the graph and irrespective of the MDP's reward structure. In the case of the graph Laplacian, this property ensures the eigenfunctions associated with small eigenvalues are smooth. However, it may well be the case that the value function, for certain domains, contains discontinuities. For example, it is well known that in the mountain car domain 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 many 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). The graph's edge weights, which were originally set based on topological distance, can include another term to account for smoothness in the value function. This is formalized by adjusting the edge weights as follows

$$W(x,y) = e^{-\frac{\|x-y\|^2}{\sigma_1} - \frac{|f(x)-f(y)|^2}{\sigma_2}}.$$

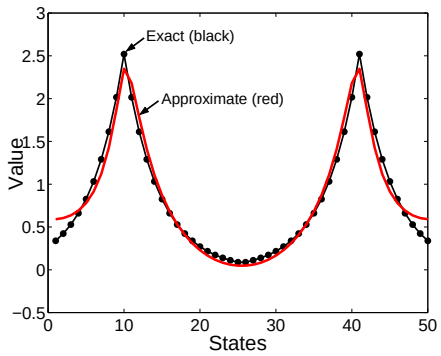
The first term in the equation controls for the distance between data points while the second term controls for the distance between functions evaluated at the data points. In the case of MDPs, the function f could be the estimated value function. The parameters σ_1 and σ_2 can be set according to the 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 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.

The most obvious way to implement this 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 will have to be recomputed at this point which can be expensive, although there may be ways to reuse previous intermediate computation. The learning algorithm can then be rerun with the new basis functions. It is important to point out that although this extension to include reward information in the graph is fairly straightforward, our previous work was always done independent of the reward which allows the bases to be reused for various learning tasks. The flexibility to be either policy dependent or independent is a strength of the graph based framework.

To demonstrate the effectiveness of this change of representation, we present results using the 50-state open chain MDP. This is a domain with 50 states with a positive reward only in states 10 and 41, and zero reward otherwise. The discount factor was set to $\gamma = 0.8$. Figure 8(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 8(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

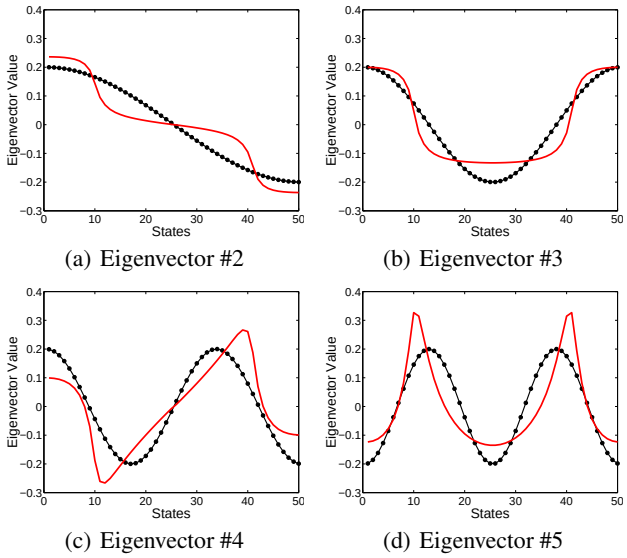


(b) Approximation with data and function dependent basis functions

Figure 8: 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. Eigenvectors 2-5 are shown in Figure

9. 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) Eigenvector #2 (b) Eigenvector #3

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

Figure 9: 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).

Basis Function Adaptation to 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 ideas that are unique to diffusion wavelets. As we 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 used. For example, Maggioni and Mahadevan (2006) demonstrated that diffusion wavelets, which efficiently represent powers of the transition matrix, can be used to quickly invert the transition matrix. 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. Diffusion wavelets could be used in conjunction with the transition matrix to automatically generate options.

Conclusions and Future Work

In this paper, we reviewed the RPI algorithm. This algorithm allows not only a flexible framework for automatic basis constructon, but can also adapt the representation based upon 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 have 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 more insight to basis construction. Currently we have 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.

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 (to appear)*. 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.; 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.

Osentoski, S., and Mahadevan, S. 2007. Learning state-action basis functions for hierarchical MDPs. In *Proceedings of the 24th International Conference on Machine Learning (to appear)*. 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.

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

Smart, W. 2004. Explicit manifold representations for value-function approximation in reinforcement learning. In *Proceedings of the 8th International Symposium on AI and Mathematics*.

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.