Constructing Basis Functions from Directed Graphs for Value Function Approximation

Jeff Johns JOHNS@CS.UMASS.EDU
Sridhar Mahadevan MAHADEVA@CS.UMASS.EDU
Dept. of Computer Science, Univ. of Massachusetts Amherst, 140 Governors Dr., Amherst, MA 01003 U.S.A.

Abstract

Basis functions derived from an undirected graph connecting nearby samples from a Markov decision process (MDP) have proven useful for approximating value functions. The success of this technique is attributed to the smoothness of the basis functions with respect to the state space geometry. This paper explores the properties of bases created from *directed* graphs which are a more natural fit for expressing state connectivity. Digraphs capture the effect of non-reversible MDPs whose value functions may not be smooth across adjacent states. We provide an analysis using the Dirichlet sum of the directed graph Laplacian to show how the smoothness of the basis functions is affected by the graph's invariant distribution. Experiments in discrete and continuous MDPs with non-reversible actions demonstrate a significant improvement in the policies learned using directed graph bases.

1. Introduction

Function approximation is a critical component to solving Markov decision processes defined over continuous state spaces or large discrete state spaces. A useful function approximator allows a reinforcement learning agent to not only accurately represent a value for a state it has experienced, but also generalize values to nearby states it has not experienced. The most common technique is linear function approximation where value functions are represented as a linear combination of basis functions. Radial basis functions, polynomial state encodings, and CMACs are examples of basis functions typically used (Sutton & Barto, 1998). Parameters associated with these basis functions can be tuned to a specific value function (Menache et al., 2005). These basis functions can be effective but require significant, domain specific hand-engineering. To address this challenge, several methods have recently been proposed to automatically construct basis functions. These methods can be categorized as either policy dependent (Keller et al., 2006; Petrik, 2007) or policy independent (Smart, 2004; Mahadevan, 2005).

Keller et al. (2006) used the Bellman error to guide the mapping from a high dimensional state space to a low dimensional space. Basis functions were created by aggregating states in the low dimensional space. By adding new basis functions tuned to the current Bellman error, this approach adaptively adjusts the basis subspace to represent more of the approximate value function. Petrik (2007) used the probability transition function and the reward model to create basis functions from Krylov space vectors (Golub & Van Loan, 1996). Both techniques assume the probability transition function and reward model are known or can be estimated from sample trajectories. Furthermore, they did not attempt to solve the control problem.

In contrast, the aim of policy independent basis functions is to capture intrinsic domain structure which should be useful for representing any *smooth* value function. Smart (2004) proposed using manifolds to model the topology of the state space. Charts of a set size were allocated to cover the state space, where each chart's embedding function provided the bases for representing a value function. Mahadevan (2005) also proposed modeling the state space topology as a manifold, but used graphs instead of charts as the computational framework. These two frameworks are related because the discrete graph Laplacian studied in spectral graph theory (Chung, 1997) approximates the Laplace-Beltrami operator on continuous manifolds. The state space connectivity is modeled using an undirected graph represented as a weight matrix W. Note

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

W is **not** estimating transition probabilities which requires many samples. W is used to form either the combinatorial or normalized graph Laplacian whose low order eigenvectors (i.e. those associated with the smallest eigenvalues) are used as basis functions. The bases are well-suited to represent smooth value functions (where smoothness is a byproduct of the Bellman equation) because the Laplacian eigenvectors are the *smoothest* functions defined over the graph and capture nonlinearities in the domain.

This paper builds on the policy independent Laplacian framework, but considers the recently introduced directed graph Laplacian (Chung, 2005). Directed graphs can better reflect non-reversible actions. This is an important property because value functions are only smooth with respect to the dynamics of the environment. A simple example to illustrate this point is a cyclic graph where actions only move an agent in one direction and a reward exists at one goal state. The goal state and its subsequent state will have different values due to the non-reversibility. In this paper, we compare the performance of policies learned using the directed versus the undirected Laplacian basis functions in a discrete and continuous MDP. Results suggest the directed Laplacian bases provide a better basis subspace for value function approximation. We analyze this finding in terms of the Dirichlet sum of the Laplacian.

2. MDPs and Least-Squares Approximation

We briefly review MDPs and least-squares approximation techniques. 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:

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

For large MDPs, an exact representation of the value function is intractable. A linear function approximation scheme is often employed to represent the value function with a set of k basis functions ($k \ll |S| \times |A|$) $\phi(s,a)$

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

where the weights $\mathbf{w}$ are tuned to minimize the error in the approximation. 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. The least squares policy iteration (LSPI) algorithm (Lagoudakis & Parr, 2003) uses a linear approximation scheme that attempts to find a fixed point of the Bellman equation $T_\pi Q^\pi \approx Q^\pi$, where T_π is the Bellman operator. The fixed point solution for the weight vector is

$$\mathbf{w}^\pi = (\Phi^T(\Phi - \gamma P \Pi_\pi \Phi))^{-1} \ (\Phi^T R) = A^{-1} b,$$

where γ is the discount factor and $P\Pi_\pi$ is the transition matrix defined with respect to policy π. The weights $\mathbf{w}^\pi$ minimize the distance between the approximate action value function and the function's projection onto the subspace spanned by the basis functions. The algorithm iterates until the weight vector converges within a specified threshold. Lastly, since the dynamics ($P^a_{ss'}$ and $R^a_{ss'}$) are unknown, the LSPI algorithm uses estimates based on the samples in the training set. The matrix $\hat{A}$ and vector $\hat{b}$ are calculated from samples via the update equations

$$\hat{A}^{(t+1)} = \hat{A}^t + \phi(s_t,a_t)(\phi(s_t,a_t) - \gamma\phi(s'_t, \pi(s'_t)))^T$$
$$\hat{b}^{(t+1)} = \hat{b}^t + \phi(s_t,a_t) r_t.$$

The Representation Policy Iteration (RPI) algorithm (Mahadevan, 2005) uses the LSPI method but constructs the basis functions automatically from the samples. RPI builds a graph where the vertices represent states and the edges correspond to the local neighborhood relation. A spectral analysis of the graph Laplacian (Chung, 1997) generates the k smoothest eigenfunctions which in turn are used as the basis functions for approximating the value function.

3. Directed Graph Laplacian

Consider an undirected graph $G = (V, E, W)$ with n vertices V, edges E, and symmetric positive weights W for each edge $(i,j) \in E$. W can be represented as a matrix with 0 values indicating lack of edges in the graph. The valency matrix D is a diagonal matrix whose values are the row sums of W ($D_{ii} = \sum_k W_{ik}$). The combinatorial graph Laplacian (Chung, 1997) is defined as $L_u = D - W$ and the normalized graph Laplacian is $\mathscr{L}_u = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. Since W is symmetric, the eigenvectors of the Laplacian form a complete orthonormal basis.

The definition of the graph Laplacian was recently generalized to the case of directed graphs (Chung, 2005) whose weight matrices can be asymmetric. Before giving the definitions, two more matrices need to be mentioned. First, $P = D^{-1}W$ is the probability transition

matrix associated with a random walk on the directed graph. There are multiple ways to handle vertices with out-degree 0 such that P is well-defined. In this paper, we use the convention that 0 out-degree vertices have a uniform $\frac{1}{n}$ probability of transitioning to any state. Second, Ψ is a diagonal matrix with $\Psi_{ii} = \psi_i$, where ψ is the Perron vector of P. The Perron-Frobenius theorem ensures that if the directed graph is *strongly connected* and *aperiodic*, then P has a unique left eigenvector ψ with all positive entries such that $\psi^T P = \rho\psi^T$ and all other eigenvalues have absolute value less than ρ. Since P is stochastic, the spectral radius $\rho = 1$ and ψ can be normalized such that $\sum_i \psi_i = 1$. The vector ψ is the invariant distribution upon convergence of a random walk on the digraph. The combinatorial (L_d) and normalized ($\mathscr{L}_d$) directed Laplacians are then defined as

$$L_d = \Psi - \frac{\Psi P + P^T\Psi}{2} \quad (1)$$

$$\mathscr{L}_d = I - \frac{\Psi^{1/2} P \Psi^{-1/2} + \Psi^{-1/2} P^T \Psi^{1/2}}{2}. \quad (2)$$

The undirected Laplacian is a special case of the directed Laplacian. This claim is easily verified by inputting a symmetric weight matrix into the equations for the directed Laplacian and simplifying (there is an added constant due to the normalization constraint on ψ). Both directed Laplacians are symmetric matrices, ensuring a complete orthonormal basis of real eigenvectors. The symmetrization $\frac{1}{2}(\Psi P + P^T\Psi)$ essentially creates an undirected graph with edge weights $\frac{1}{2}(\psi_i P_{ij} + \psi_j P_{ji}) = \frac{1}{2}(\frac{\psi_i}{D_{ii}} W_{ij} + \frac{\psi_j}{D_{jj}} W_{ji})$. Mahadevan et al. (2006) proposed using three different symmetrization techniques: $\frac{1}{2}(W + W^T)$, WW^T, and W^TW. Their first method, $\frac{1}{2}(W + W^T)$, creates an undirected graph with edge weights $\frac{1}{2}(W_{ij} + W_{ji})$. We will compare empirically and analytically these two techniques: $\frac{1}{2}(\Psi P + P^T\Psi)$ versus $\frac{1}{2}(W + W^T)$.

Although we focus here on forming symmetric matrices, it is also possible to consider asymmetric Laplacian matrices (Agaev & Chebotarev, 2005) which can have complex eigenvalues. Asymmetric matrices may not be diagonalizable but can nevertheless be decomposed into Jordan normal form (e.g. all non-zero entries are on the diagonal and superdiagonal); however, these algorithms are numerically unstable.

3.1. Calculating the Perron Vector

The properties guaranteed by the Perron-Frobenius theorem depended on the directed graph being strongly connected and aperiodic (note the theorem still holds if the graph is strongly connected and *periodic*, but the first eigenvalue is no longer simple). A MDP need not have these properties. One way to ensure these two properties are met is to assume a teleporting random walk (Page et al., 1998). The agent acts according to the transition matrix P with probability η and with probability $(1 - \eta)$ teleports to any vertex uniformly at random. This assumption is used only for the purpose of creating ψ and performing the spectral decomposition; the agent is not allowed to teleport in the MDP. The teleporting transition matrix is defined as

$$P_{teleport} = \eta P + (1 - \eta)\frac{\mathbf{1}\mathbf{1}^T}{n}$$

where $\mathbf{1}$ is a column vector of ones of length n. There is a trade-off in setting η: larger values allow $P_{teleport}$ to be more similar to P, but smaller values can increase the eigengap which is useful for computational and stability reasons when performing spectral analysis.

$P_{teleport}$ should not be explicitly formed when implementing this technique since it is a full matrix while P is typically sparse. Computing the eigenvectors of L_d or $\mathscr{L}_d$ only requires access to a method for computing the matrix-vector product $L_d\mathbf{x}$ or $\mathscr{L}_d\mathbf{x}$; thus, the method can handle the two components of $P_{teleport}$ separately. To see how this is useful, consider multiplication of the second component by a vector $\mathbf{x}$: $\frac{1-\eta}{n}(\mathbf{1}\,\mathbf{1}^T)\mathbf{x} = \frac{1-\eta}{n}(\sum_i x_i)\mathbf{1}$. The right-hand side of the equation is computed in $O(n)$ without forming the matrix of ones. The eigendecompositions of the undirected and directed Laplacians are therefore computationally equivalent.

The only added cost for the directed Laplacian is in calculating ψ. We used the power method (Golub & Van Loan, 1996) which is an iterative technique based on the definition $\psi^T P_{teleport} = \psi^T$. Starting with an initial guess for ψ, the vector is multiplied by $P_{teleport}$ to get a new estimate. The process repeats until ψ converges within a specified tolerance. The convergence rate depends on the gap between the first and second eigenvalues. The power method converged in less than one second in our experiments on graphs with fewer than 1000 vertices. Although scalability is an issue for very large graphs, several algorithms have been proposed to speed up this computation (this effort is attributable to the success of Google's PageRank$^{\text{TM}}$that calculates ψ for the web graph).

3.2. Related Work

The directed Laplacian has been applied by Zhou et al. (2005) to perform semi-supervised learning in the context of web page classification. The Rayleigh quotient was shown to enforce smoothness of the labels assigned

to the vertices. Experiments showed a significant improvement in accuracy by using a directed graph compared to an undirected graph. This paper investigates whether directionality plays a similarly important role for value function approximation.

4. Experiments

Experiments were conducted in discrete and continuous domains. Samples were generated from the domains using a random policy, but more intelligent forms of exploration could also be used. A directed graph was constructed from the samples where vertices correspond to state variables. The digraph construction followed along the lines discussed in (Mahadevan et al., 2006) with a slight difference for the continuous domain (discussed in detail below). In the discrete domain, directed edges were added for actual state transitions seen in the training episodes. In the continuous domain, the definition of directed edges was generalized to respect the effects of actions. After building the graph, the combinatorial Laplacian was formed using the two types of symmetrization, $\frac{1}{2}(\Psi P + P^T \Psi)$ and $\frac{1}{2}(W + W^T)$, with a probability of not teleporting $\eta = 0.99$. Note the Laplacian basis functions correspond to state embeddings $\phi(s)$; these embeddings were copied for each action yielding $\phi(s, a)$. A policy was learned using the RPI algorithm and then evaluated to compare the two symmetrizations.

The next two sections provide domain specific details.

4.1. Discrete Grid

We tested a 200 state grid world MDP shown in Figure 1, where all actions are reversible except for the two directed doorways which connect the 10×10 rooms. The states S_A and S_B are labeled in the figure for ease of exposition throughout the paper. The four directional actions are stochastic. An action is successful with probability 0.9, whereas a failed action results in the agent transitioning (uniformly at random) according to one of the other three actions. The agent receives a reward of $+100$ upon reaching the goal state S_G and 0 otherwise. The discount factor was set to $\gamma = 0.95$. Although there are only two non-reversible actions in this domain, they are in key locations which give the agent access to large regions of the state space. The optimal value function for this domain is shown in Figure 2. In particular, notice the large discontinuity in the value function from state S_A to S_B.

The RPI learning algorithm was run with samples from 800 training episodes. Each episode began with the agent in a random state. The agent followed a random policy that terminated either upon reaching S_G or after 20 steps were exceeded. 800 episodes were chosen because this often resulted in each state being visited at least once. We varied the number of basis functions the RPI algorithm used from 5 to 50. Since more complex functions can be represented given a larger basis set, the experimental results allow comparison of the two symmetrizations with varying degrees of basis space complexity.

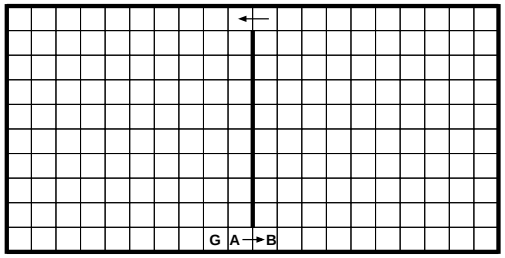


Figure 1. Directed two room grid MDP where the arrows represent non-reversible actions. S_G is the goal state. States S_A and S_B are labeled for ease of exposition.

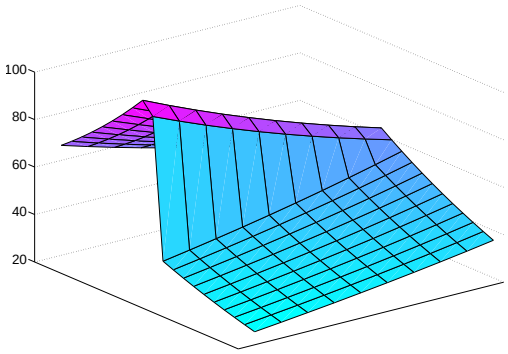


Figure 2. Optimal value function for the grid MDP.

4.2. Inverted Pendulum

The inverted pendulum is a continuous control problem consisting of a pendulum attached to a cart. The goal is to learn a policy for balancing the pendulum by applying force to the cart. The state space can be modeled using two dimensions: θ, the vertical angle of the pendulum, and $\dot{\theta}$, the angular velocity of the pendulum. There are three actions corresponding to a force of -50, 0, and $+50$ Newtons with additive noise generated from a uniform distribution over the range -10 to $+10$. The dynamics are governed by the following nonlinear equation

$$\ddot{\theta} = \frac{g \sin(\theta) - \frac{1}{2}\alpha m l \dot{\theta}^2 \sin(2\theta) - \alpha \cos(\theta) u}{\frac{4}{3} l - \alpha m l \cos^2(\theta)}$$

where u is the noisy action, $g = 9.8\ m/s^2$ is gravity, $m = 2\ kg$ is the mass of the pendulum, $M = 8\ kg$ is the mass of the cart, $l = 0.5\ m$ is the length of the pendulum, and $\alpha = 1/(m+M)$. The simulation time step was set to 0.1 seconds. The episode ends and the agent receives a reward of -1 when $|\theta| \geq \pi/2$. All other actions result in a reward of 0. The discount factor was set to $\gamma = 0.95$. There are many non-reversible actions that occur when θ exceeds a tipping point from which none of the actions can keep the pendulum upright.

In contrast to the discrete MDP experiments, the inverted pendulum experiments were run with a fixed number of basis functions (8) and a varying number of training episodes (5 to 150). We did this because policies tended to be more binary in nature, either learned perfectly or not at all. Only eight basis functions were used because of the value function's simplicity. Each training episode began with the agent in state $[\theta,\ \dot{\theta}] = [0,\ 0]$. The agent followed a random policy that typically lasted ≈10 steps before the pendulum fell.

Digraph construction in continuous domains is more challenging than in discrete domains. After generating the samples, we subsampled the data using a simple greedy procedure: starting with the null set, add samples to the subset that are **not** within a specified distance (0.2 in our experiments) to any sample currently in the subset. A maximal subset is returned when no more samples can be added. We used a weighted Euclidean distance metric to compute the distance between two states. The weight was 3 for θ and 1 for $\dot{\theta}$. This produced a more equal range for the two dimensions. A graph was then constructed over the subsampled points by connecting each vertex to its $k = 25$ nearest neighbors. Each edge was assigned a weight using the Gaussian kernel with $\sigma = 1.5$ and the weighted Euclidean distance between the vertices. This is nearly identical to the procedure outlined in (Mahadevan et al., 2006) (technique $E(2)$ for the edges and $W(2)$ for the weights) except we used the weighted Euclidean distance. We then added one more step to the graph construction. Edges that do not respect the directionality of the three actions were pruned. We used cosine similarity with edges corresponding to vectors to determine if an edge was similar in direction to at least one of the actions. If the edge was outside of a specified limit (θ^*=50°), then it was removed from the graph. This results in a directed graph that captures the structure of the dynamics. The exact algorithm for this construction is shown in Figure 3. Note the vectors v and v_a in the algorithm correspond to changes in the state values. This simple approach works well in this two dimensional domain, but more robust methods should be used for high dimensional state spaces. Lastly, the embeddings for novel states not contained in the subsampled set were created using the Nyström extension (Williams & Seeger, 2001). This method interpolates based on the embeddings of "nearby" states in the subsampled set.

`Directed Graph Construction` $(S, S'_a, k, D, \sigma, \theta^*)$:
// S: States in subsample set (each state $S_i \in \Re^d$)
// S'_a: Next state upon taking action a in state S,
// one for each discrete action
// k: Number of nearest neighbors
// D: Distance metric for comparing two states
// σ: Distance parameter for assigning weights
// θ^*: Threshold for pruning edges

1. For each state $S_i \in S$, create an edge from S_i to its k nearest neighbors
2. Assign weight $e^{-\frac{D(S_i,S_j)}{\sigma}}$ to each edge
3. **Prune edges based on directionality**
   For each edge $S_i \rightarrow S_j$:
   $v = (S_j - S_i)$
   For each action a:
   $v_a = (S'_{i,a} - S_i)$
   $\theta = \min_a \ \text{acos}\left(\frac{v \cdot v_a}{\|v\|\ \|v_a\|}\right)$
   If $(\theta > \theta^*)$
   Remove edge $S_i \rightarrow S_j$

Figure 3. Pseudo-code for directed graph construction in continuous domains with discrete actions.

5. Results

The experimental results are discussed in the next two sections.

5.1. Discrete Grid

Learned policies were evaluated to determine the discounted reward given a uniform initial state distribution. Thus, a policy was tested by starting an agent once in each of the 200 states and letting it run until reaching the goal or exceeding 100 steps. The reward was averaged over the 200 different starting positions. These experiments were repeated 100 times. The median over the 100 runs is shown in Figure 4 with error bars corresponding to the first and third quartiles. Performance for both symmetrizations was poor when using less than 20 basis functions. The $\frac{1}{2}(\Psi P + P^T \Psi)$ symmetrization consistently outperformed the $\frac{1}{2}(W + W^T)$ symmetrization when using more than 20 basis functions. The first and third quartiles indicate the policies ranged widely when using less than 30 basis functions, but became more consistent when using more than 30.

By placing the goal state next to state S_A, there was a significant difference in the value function from S_A to S_B. This seems like an ideal case for the directed Laplacian to perform well. The value function would not have this discontinuity if the goal state were in the opposite room adjacent to S_B. We ran experiments with this setup and achieved similar results to those shown in Figure 4 although the gap between the curves was not quite as large. Thus, policies learned using the $\frac{1}{2}(\Psi P + P^T\Psi)$ symmetrization are better even if the value function is smooth across the doorways.

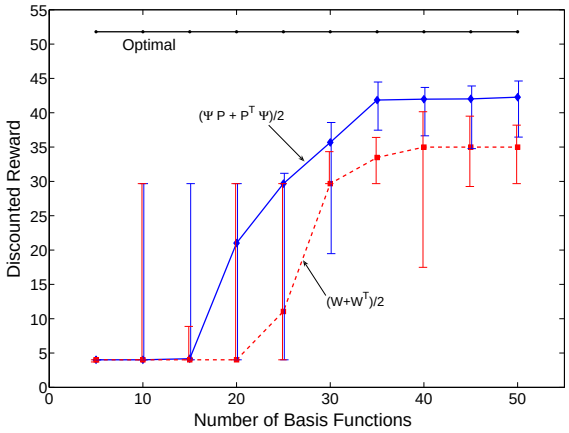


Figure 4. Median performance over 100 runs for the discounted reward in the grid domain. Error bars correspond to the first and third quartiles.

5.2. Inverted Pendulum

The experiments were repeated 50 times with a varying number of training episodes. The learned policies were tested 10 times for a maximum of 3000 steps, with the final result being the average (over 10 tests) number of balancing steps. The median over the 50 runs is shown in Figure 5 with the error bars corresponding to the first and third quartiles. The $\frac{1}{2}(\Psi P + P^T\Psi)$ symmetrization required much less experience in order to learn an effective policy. With only 50 training episodes, the policy rarely dropped the pendulum (the first quartile is at the maximum 3000 steps). On the other hand, the $\frac{1}{2}(W + W^T)$ symmetrization performed well with 150 training episodes, but with a much greater variance in performance (the first quartile is only at 200 steps).

6. Analysis

In this section, we address why the $\frac{1}{2}(\Psi P + P^T\Psi)$ symmetrization outperformed the $\frac{1}{2}(W + W^T)$ symmetrization in these two domains. The performance gain is best explained by analyzing the basis function

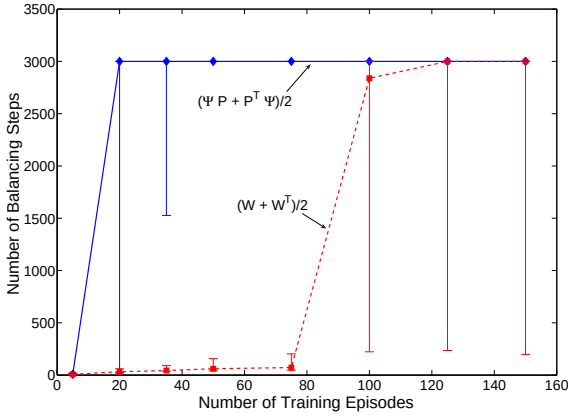


Figure 5. Median performance over 50 runs in the inverted pendulum domain. Error bars correspond to the first and third quartiles.

properties. Basis functions should be smooth with respect to the geometry of the state space because value functions are typically smooth over the state space. The smoothness of value functions is due to the Bellman equation, where the action value at a state is a linear function of the values at neighboring states.

The smoothness of a function f on a graph can be measured by the Sobolev norm

$$\begin{aligned}\|f\|^2_{\mathscr{H}^2} &= \|f\|_2^2 + \|\nabla f\|_2^2 \\ &= \sum_v |f(v)|^2 d(v) + \sum_{u\sim v} W_{uv}\left(f(u) - f(v)\right)^2.\end{aligned}$$

The second term in the Sobolev norm is the Dirichlet sum of the function, which for the Laplacian of a symmetric weight matrix is equal to $\|\nabla f\|_2^2 = \langle f, Lf\rangle$. Functions that are smooth over adjacent vertices produce a small Dirichlet sum.

The Dirichlet sum for the $\frac{1}{2}(W+W^T)$ symmetrization of a directed graph with a asymmetric weight matrix W is given in Equation 3. Equation 4 shows the Dirichlet sum for the $\frac{1}{2}(\Psi P + P^T\Psi)$ symmetrization. The major difference between the equations is that the directed Laplacian weights the squared differences by the invariant distribution ψ. This means a vertex with a large value of ψ contributes more to the Dirichlet sum than one with a small value of ψ. Thus, if the Dirichlet sum of f is to be small (i.e. f is smooth), then $f(u) \approx f(v)$ when ψ_u is large relative to other vertices and there is an edge from u to v.

$$\langle f, L_d f\rangle = \sum_{u\to v\in E} W_{uv}(f(u) - f(v))^2 \tag{3}$$

$$\langle f, L_d f\rangle = \sum_{u\to v\in E} \frac{\psi_u}{D_{uu}} W_{uv}(f(u) - f(v))^2 \tag{4}$$

We examine the grid domain in more detail to demonstrate how the Perron vector plays a significant role in defining the smoothness of the basis functions. We chose to present the results for the discrete domain instead of the continuous domain because it is easier to visualize the basis functions, but the same concepts hold in both domains. Figure 6 shows the Perron vector for the grid. There is a relatively large difference in the value at state S_A ($\psi_{S_A} = 0.0021$) versus S_B ($\psi_{S_B} = 0.0079$).

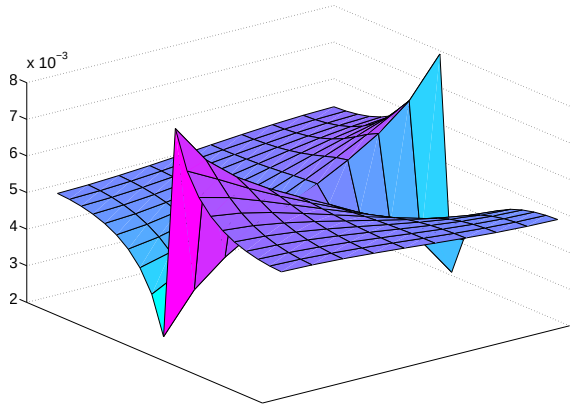
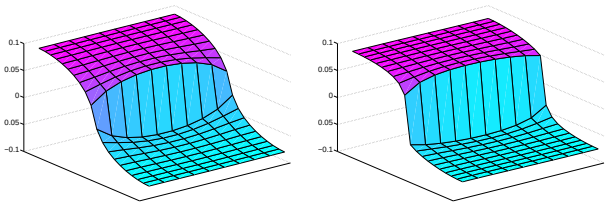


Figure 6. Perron vector, $\psi_{S_A} = 0.0021$ and $\psi_{S_B} = 0.0079$.

To visualize the effect of the Perron vector on function smoothness, we present the second eigenvector of the combinatorial Laplacian using the two different types of symmetrization in Figure 7. The second eigenvector of the Laplacian is the smoothest function orthogonal to the first eigenvector (which is simply a constant vector for the combinatorial Laplacian). Notice how state S_B in Figure 7(b) has a value very similar to both states it can reach in one step. On the other hand, state S_A in Figure 7(b) has a very different value from S_B but a similar value to the other two states it can reach in one step. This effect is not seen in Figure 7(a), where states S_A and S_B are constrained to be similar to each other. The constraint between S_A and S_B has affected the value of neighboring states. The eigenvector is clearly more curved. Although this distinction may appear small, the difference in policies derived from these basis functions can be significant.

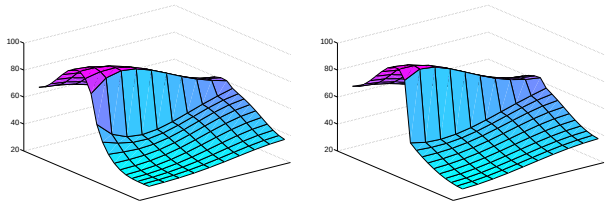
The low order basis functions of the Laplacian formed by the $\frac{1}{2}(W+W^T)$ symmetrization constrain functions to be smooth across states S_A and S_B. The low order basis functions are particularly important for a Fourier style analysis because their coefficients will be larger than the coefficients associated with the higher order basis functions. Thus, constraints introduced in the low order basis functions are likely to be manifested in the approximated function. In the grid domain, the



(a) $\frac{1}{2}(W + W^T)$ (b) $\frac{1}{2}(\Psi P + P^T \Psi)$

Figure 7. Second eigenvector of the combinatorial Laplacian with different symmetrization methods.

optimal value function has a large discontinuity from S_A to S_B. As expected, the basis functions for the $\frac{1}{2}(W + W^T)$ symmetrization have a harder time representing this gap. Figure 8 shows the optimal value function projected onto the 10 smoothest basis functions of the combinatorial Laplacian. The approximate value function in Figure 8(a) is heavily influenced by the smoothness constraint on S_A and S_B. An agent acting according to this value function makes mistakes in the states surrounding S_B (e.g. the agent follows the value function to state S_B where it becomes stuck since it cannot pass through the door).



(a) $\frac{1}{2}(W + W^T)$ (b) $\frac{1}{2}(\Psi P + P^T \Psi)$

Figure 8. Projection of the optimal value function onto the first 10 eigenvectors of the combinatorial Laplacian with different symmetrization methods.

The smoothness assumption can be tied back to the Bellman equation. State S_B will never be a direct function of S_A in the Bellman equation since there is no action taking S_B to S_A (although there may be a longer temporal dependence). Conversely, S_A may be a function of S_B depending on the reward structure of the domain. This situation only occurs when there are non-reversible actions. When there are reversible actions between pairs of states, the value function is likely to be smooth across them assuming a relatively large discount factor. Thus, assumptions of smoothness should be less restrictive when dealing with non-reversible actions.

7. Conclusions and Future Work

This paper explored the use of the recently introduced directed graph Laplacian to construct basis functions for value function approximation. The definition of the directed Laplacian corresponds to one type of weight matrix symmetrization, $\frac{1}{2}(\Psi P + P^T \Psi)$. We compared this form with the simpler $\frac{1}{2}(W + W^T)$. Results in both discrete and continuous domains indicate the directed Laplacian outperforms the simple symmetrization. This finding was explored analytically using the Dirichlet sum of the Laplacian matrices. Each vertex V_u in the Dirichlet sum is weighted by the value of the invariant distribution ψ_{V_u}. The Laplacian basis functions are *smoother* across pairs of states for which there is a directed edge from V_u to V_v and ψ_{V_u} is large. The effect on the smoothness of the basis functions was demonstrated in the discrete grid domain.

Value functions are smooth with respect to the dynamics of the environment as dictated by the Bellman equation. Thus, it is more natural to use a directed graph to model the topology of the state space for non-reversible MDPs. Value functions can have discontinuities across pairs of states with non-reversible actions. The smoothness assumption should be less restrictive for these pairs.

We plan to explore properties of the basis functions with respect to the Bellman equation. The invariant distribution of the random walk helped to define a better basis space in these experiments, but a theoretical analysis would make this relationship more clear.

One of the main challenges for the Laplacian framework is scalability to large MDPs. Recent work (Johns et al., 2007) has proposed to address this issue by factorizing the transition matrix P into smaller matrices using the Kronecker product. The basis functions can then be represented more compactly on the smaller matrices. This approach can work with the directed graph Laplacian. We are also exploring matrix sparsification as another route to scaling.

8. Acknowledgements

We thank Sarah Osentoski and the anonymous reviewers for valuable comments that improved the paper. This research was funded in part by the National Science Foundation under grant NSF IIS-0534999.

References

Agaev, R., & Chebotarev, P. (2005). On the spectra of nonsymmetric Laplacian matrices. *Linear Algebra and Its Applications*, *399*, 157–168.

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

Chung, F. (2005). Laplacians and the Cheeger inequailty for directed graphs. *Annals of Combinatorics*, *9*, 1–19.

Golub, G., & Van Loan, C. (1996). *Matrix Computations*. Baltimore, MD: Johns Hopkins Univ. Press. 3rd edition.

Johns, J., Mahadevan, S., & Wang, C. (2007). Compact spectral bases for value function approximation using Kronecker factorization. *Proceedings of the 22nd National Conference on Artificial Intelligence*.

Keller, P., Mannor, S., & Precup, D. (2006). Automatic basis function construction for approximate dynamic programming and reinforcement learning. *Proceedings of the 23rd International Conf. on Machine Learning* (pp. 449–456).

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

Mahadevan, S. (2005). Representation policy iteration. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence* (pp. 372–379).

Mahadevan, S., Maggioni, M., Ferguson, K., & Osentoski, S. (2006). Learning representation and control in continuous Markov decision processes. *Proceedings of the 21st National Conference on Artificial Intelligence*.

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

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). *The PageRank citation ranking: Bringing order to the web* (Technical Report). Stanford Digital Library Technologies Project.

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

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

Sutton, R., & Barto, A. (1998). *Reinforcement learning*. Cambridge, MA: MIT Press.

Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems 13* (pp. 682–688).

Zhou, D., Huang, J., & Schölkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning* (pp. 1036–1043).