Value Function Approximation in Reinforcement Learning using the Fourier Basis

George Konidaris[1,3]
MIT CSAIL[1]
gdk@csail.mit.edu

Sarah Osentoski[2,3*]
Department of Computer Science[2]
Brown University
sosentos@cs.brown.edu

Philip Thomas[3]
Autonomous Learning Laboratory[3]
University of Massachusetts Amherst
pthomas@cs.umass.edu

Abstract

We describe the Fourier basis, a linear value function approximation scheme based on the Fourier series. We empirically demonstrate that it performs well compared to radial basis functions and the polynomial basis, the two most popular fixed bases for linear value function approximation, and is competitive with learned proto-value functions.

Introduction

Reinforcement learning (RL) in continuous state spaces requires function approximation. Most work in this area focuses on *linear function approximation*, where the value function is represented as a weighted linear sum of a set of features (known as basis functions) computed from the state variables. Linear function approximation results in a simple update rule and quadratic error surface, even though the basis functions themselves may be arbitrarily complex.

RL researchers have employed a wide variety of basis function schemes, most commonly radial basis functions (RBFs) and CMACs (Sutton and Barto, 1998). Often, choosing the right basis function set is criticial for successful learning. Unfortunately, most approaches require significant design effort or problem insight, and no basis function set is both simple and sufficiently reliable to be generally satisfactory. Recent work (Mahadevan and Maggioni, 2007) has focused on learning basis function sets from experience, removing the need to design an approximator but introducing the need to gather data to create one.

The most common continuous function approximation method in the applied sciences is the Fourier series. Although the Fourier series is simple, effective, and has solid theoretical underpinnings, it is almost never used for value function approximation. This paper describes the Fourier basis, a simple linear function approximation scheme using the terms of the Fourier series as basis functions.[1] We demonstrate that it performs well compared to RBFs and the polynomial basis, the most common fixed bases, and is competitive with learned proto-value functions even though no extra experience or computation is required.

*Sarah Osentoski is now with the Robert Bosch LLC Research and Technology Center in Palo Alto, CA.

[1]Open-source, RL-Glue (Tanner and White, 2009) compatible Java source code for the Fourier basis can be downloaded from http://library.rl-community.org/wiki/Sarsa_Lambda_Fourier_Basis_(Java).

Background

A d-dimensional continuous-state Markov decision process (MDP) is a tuple $M = (S, A, P, R)$, where $S \subseteq \mathbb{R}^d$ is a set of possible state vectors, A is a set of actions, P is the transition model (with $P(\mathbf{x}, a, \mathbf{x}')$ giving the probability of moving from state $\mathbf{x}$ to state $\mathbf{x}'$ given action a), and R is the reward function (with $R(\mathbf{x}, a, \mathbf{x}')$ giving the reward obtained from executing action a in state $\mathbf{x}$ and transitioning to state $\mathbf{x}'$). Our goal is to learn a policy, π, mapping state vectors to actions so as to maximize return (discounted sum of rewards). When P is known, this can be achieved by learning a *value function*, V, mapping state vectors to return, and selecting actions that result in the states with highest V. When P is not available, the agent will typically either learn it, or instead learn an *action-value function*, Q, that maps state-action pairs to expected return. Since the theory underlying the two cases is similar we consider only the value function case. V is commonly approximated as a weighted sum of a set of basis functions $\phi_1, ..., \phi_m$: $\bar{V}(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$. This is termed *linear value function approximation* since $\bar{V}$ is linear in weights $\mathbf{w} = [w_1, ..., w_m]$; learning entails finding the $\mathbf{w}$ corresponding to an approximate optimal value function, $\bar{V}^*$. Linear function approximation is attractive because it results in simple update rules (often using gradient descent) and possesses a quadratic error surface with a single minimum (except in degenerate cases). Nevertheless, we can represent complex value functions since the basis functions themselves can be arbitrarily complex.

The Polynomial Basis. Given d state variables $\mathbf{x} = [x_1, ..., x_d]$, the simplest linear scheme uses each variable directly as a basis function along with a constant function, setting $\phi_0(\mathbf{x}) = 1$ and $\phi_i(\mathbf{x}) = x_i$, $0 \leq i \leq d$. However, most interesting value functions are too complex to be represented this way. This scheme was therefore generalized to the polynomial basis (Lagoudakis and Parr, 2003): $\phi_i(\mathbf{x}) = \prod_{j=1}^{d} x_j^{c_{i,j}}$, where each $c_{i,j}$ is an integer between 0 and n. We describe such a basis as an order n polynomial basis. For example, a 2nd order polynomial basis defined

over two state variables x and y would have feature vector: $\mathbf{\Phi} = [1, \quad x, \quad y, \quad xy, \quad x^2y, \quad xy^2, \quad x^2y^2]$. Note the features that are a function of both variables; these features model the interaction between those variables.

Radial Basis Functions. Another common scheme is RBFs, where each basis function is a Gaussian: $\phi_i(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-||c_i - \mathbf{x}||^2/2\sigma^2}$, for a given collection of centers c_i and variance σ^2. The centers are typically distributed evenly along each dimension, leading to n^d centers for d state variables and a given order n; σ^2 can be varied but is often set to $\frac{2}{n-1}$. RBFs only generalize locally—changes in one area of the state space do not affect the entire state space. Thus, they are suitable for representing value functions that might have discontinuities. However, this limited generalization is often reflected in slow initial performance.

Proto-Value Functions. Recent research has focused on learning basis functions given experience. The most prominent learned basis is *proto-value functions* or PVFs (Mahadevan and Maggioni, 2007). In their simplest form, an agent builds an adjacency matrix, A, from experience and then computes the Laplacian, $L = (D - A)$, of A where D is a diagonal matrix with $D(i, i)$ being the out-degree of state i. The eigenvectors of L form a set of bases that respect the topology of the state space (as reflected in A), and can be used as a set of orthogonal bases for a discrete domain.

Mahadevan et al. (2006) extended PVFs to continuous domains, using a local distance metric to construct the graph and an out-of-sample method for obtaining the values of each basis function at states not represented in A. Although the given results are promising, PVFs in continuous spaces require samples to build A, an eigenvector decomposition to build the basis functions, and pose several potentially difficult design decisions.

The Fourier Basis

In this section we describe the Fourier series for one and multiple variables, and use it to define the univariate and multivariate Fourier bases.

The Univariate Fourier Series

The Fourier series is used to approximate a periodic function; a function f is periodic with period T if $f(x + T) = f(x), \forall x$. The nth degree Fourier expansion of f is:

$$\bar{f}(x) = \frac{a_0}{2} + \sum_{k=1}^{n} \left[a_k \cos\left(k\frac{2\pi}{T}x\right) + b_k \sin\left(k\frac{2\pi}{T}x\right) \right], \tag{1}$$

with $a_k = \frac{2}{T}\int_0^T f(x)\cos(\frac{2\pi kx}{T})\,dx$ and with $b_k = \frac{2}{T}\int_0^T f(x)\sin(\frac{2\pi kx}{T})\,dx$. For the remainder of this paper we assume for simplicity that $T = 2$, with the variables of the function we wish to approximate scaled appropriately.

In the RL setting f is unknown so we cannot compute $a_0, ..., a_n$ and $b_1, ..., b_n$, but we can instead treat them as parameters in a linear function approximation scheme, with:

$$\phi_i(x) = \begin{cases} 1 & i = 0 \\ \cos\left(\frac{i+1}{2}\pi x\right) & i > 0, i \text{ odd} \\ \sin\left(\frac{i}{2}\pi x\right) & i > 0, i \text{ even.} \end{cases}$$

Thus a full nth order Fourier approximation to a one-dimensional value function results in a linear function approximator with $2n + 1$ terms. However, as we shall see below, we can usually use only $n + 1$ terms.

Even, Odd and Non-Periodic Functions

If f is known to be even (that is, $f(x) = f(-x)$, so that f is symmetric about the y-axis), then $\forall i > 0, b_i = 0$, so the $\sin$ terms can be dropped. This results in a function *guaranteed* to be even, and reduces the terms required for an nth order Fourier approximation to $n + 1$. Similarly, if f is known to be odd (that is, $f(x) = -f(-x)$, so that f is symmetric with respect to the origin) then $\forall i > 0, a_i = 0$, so we can omit the $\cos$ terms. These cases are depicted in Figure 1.

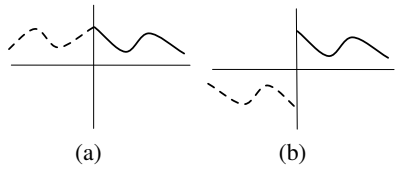


Figure 1: Even (a) and odd (b) functions.

However, in general, value functions are not even, odd, or periodic (or known to be in advance). In such cases, we can define our approximation over $[-1, 1]$ but only project the input variable to $[0, 1]$. This results in a function periodic on $[-1, 1]$ but unconstrained on $(0, 1]$. We are now free to choose whether or not the function is even or odd over $[-1, 1]$ and can drop half of the terms in the approximation.

In general, we expect that it will be better to use the "half-even" approximation and drop the $\sin$ terms because this causes only a slight discontinuity at the origin. Thus, we define the univariate nth order Fourier basis as:

$$\phi_i(x) = \cos(i\pi x), \tag{2}$$

for $i = 0, ..., n$. Figure 2 depicts a few of the resulting basis functions. Note that frequency increases with i; thus, high order basis functions will correspond to high frequency components of the value function.

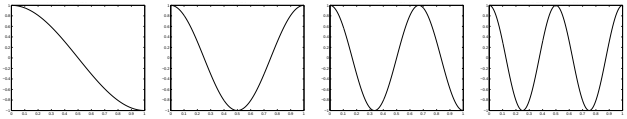


Figure 2: Univariate Fourier basis functions for $i = 1$, 2, 3 and 4. The basis function for $i = 0$ is a constant.

The Multivariate Fourier Series

The nth order Fourier expansion of the multivariate function $F(\mathbf{x})$ with period T in d dimensions is:

$$\bar{F}(\mathbf{x}) = \sum_{\mathbf{c}} \left[a_{\mathbf{c}} \cos\left(\frac{2\pi}{T}\mathbf{c}\cdot\mathbf{x}\right) + b_{\mathbf{c}} \sin\left(\frac{2\pi}{T}\mathbf{c}\cdot\mathbf{x}\right) \right], \tag{3}$$

where $\mathbf{c} = [c_1, ..., c_d]$, $c_j \in [0, ..., n]$, $1 \leq j \leq d$. This results in $2(n+1)^d$ basis functions for an nth order full Fourier approximation to a value function in d dimensions, which can be reduced to $(n+1)^d$ if we drop either the $\sin$ or $\cos$ terms for each variable as described above. We thus define the nth order Fourier basis for d variables:

$$\phi_i(\mathbf{x}) = \cos\left(\pi \mathbf{c}^i \cdot \mathbf{x}\right), \tag{4}$$

where $\mathbf{c}^i = [c_1, ..., c_d]$, $c_j \in [0, ..., n]$, $1 \leq j \leq d$. Each basis function has a vector $\mathbf{c}$ that attaches an integer coefficient (less than or equal to n) to each variable in $\mathbf{x}$; the basis set is obtained by systematically varying these coefficients.

Example basis functions over two variables are shown in Figure 3. Note that $\mathbf{c} = [0, 0]$ results in a constant function. When $\mathbf{c} = [0, k_y]$ or $[k_x, 0]$ for positive integers k_x and k_y, the basis function depends on only one of the variables, with the value of the non-zero component determining frequency. Only when $\mathbf{c} = [k_x, k_y]$ does it depend on both; this basis function represents an interaction between the two state variables. The ratio between k_x and k_y describes the direction of the interaction, while their values determine the basis function's frequency along each dimension.

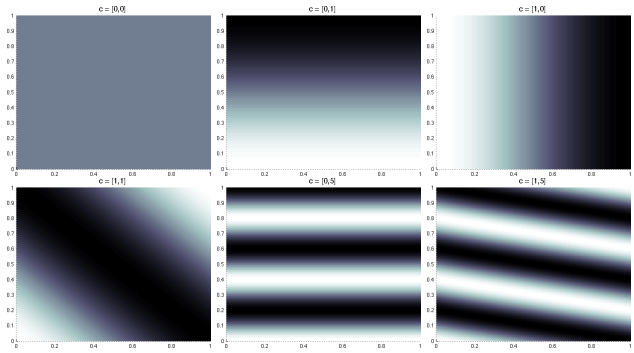


Figure 3: A few example Fourier basis functions defined over two state variables. Lighter colors indicate a value closer to 1, darker colors indicate a value closer to -1.

This basis is easy to compute accurately even for high orders, since $\cos$ is bounded in $[-1, 1]$, and its arguments are formed by multiplication and summation rather than exponentiation. Although the Fourier basis seems like a natural choice for value function approximation, we know of very few instances (e.g., Kolter and Ng (2007)) of its use in RL prior to this work, and no empirical study of its properties.

Scaling Gradient Descent Parameters

When performing stochastic gradient descent with a linear approximator, each feature need not have the same learning rate. Intuitively, lower frequency terms should have larger learning rates so that the agent can rapidly estimate the general shape of the target surface, with slower refinement by higher order terms. Many online schemes exist for tuning each learning rate, though these result in problem specific learning rates (Aleida et al., 1998). Given a base learning rate, α_1, in practice we find that setting the learning rate for basis function ϕ_i to $\alpha_i = \alpha_1 / ||\mathbf{c}^i||_2$ (avoiding division by zero by setting $\alpha_0 = \alpha_1$ where $\mathbf{c}^0 = \mathbf{0}$) performs best. An argument for this approach is given in the Appendix.

Variable Coupling

In continuous RL domains it is common to use an approximation scheme that assumes each variable contributes independently to the value function. For example, an order 1 polynomial basis function contains basis functions 1, x_1, x_2 and $x_1 x_2$; assuming independent contributions means we can drop the $x_1 x_2$ term. We call such a basis *uncoupled*.

For higher order function approximators, uncoupling removes the curse of dimensionality: for an order n polynomial basis with d variables, the full basis has $(n+1)^d$ terms, whereas the uncoupled basis has only $dn + 1$ terms. Although this can lead to poor approximations (because in many cases the variables do not contribute independently) in many continuous domains it does not significantly degrade performance. However, a more sophisticated approach could use prior knowledge about which variables are likely to interact to obtain a smaller but still accurate function approximator. The Fourier basis facilitates this through constraints on $\mathbf{c}$, the coefficient vector used in Equation 4. For example, we obtain an uncoupled basis by requiring that only one element of each $\mathbf{c}$ is non-zero. Alternatively, if we know that the interaction of variables x_i and x_j is important then we can constrain $\mathbf{c}$ so that only c_i and c_j can be non-zero simultaneously. We expect that most domains in which a decoupled basis performs poorly are actually *weakly coupled*, in that we can use very low-order terms for the interaction between variables and high-order order terms for each variable independently. The Fourier basis lends itself naturally to implementing such a scheme.

Empirical Evaluation

This section empirically compares the performance of the Fourier basis[2] on standard continuous benchmarks to that of RBFs, the polynomial basis, and PVFs. Since we are comparing basis functions (as opposed to learning algorithms), we have selected the learning algorithms most widely used, rather than those that are most efficient. While better performance might be obtained using other learning methods, we expect the relative performance differences to persist.

The Swing-Up Acrobot. The acrobot is an underactuated two-link robot where the first link is suspended from a point and the second can exert torque. The goal is to swing the tip of the acrobot's second joint a segment's length above the level of its suspension, much like a gymnast hanging from a pole and swinging above it using their waist. Since the acrobot is underactuated it must swing back and forth to gain momentum. The resulting task has 4 continuous variables (an angle and an angular velocity for each joint) and three actions (exerting a negative, positive or zero unit of torque on the middle joint). Sutton and Barto (1998) contains a more detailed description.

We employed Sarsa(λ) ($\gamma = 1.0$, $\lambda = 0.9$, $\epsilon = 0$) with Fourier bases of orders 3 (256 basis functions), 5 (1296 ba-

[2]Our experiments used scaled α values. For results using unscaled α, see Konidaris and Osentoski (2008).

sis functions) and 7 (4096 basis functions) and RBF, polynomial and PVF bases of equivalent sizes (we did not run PVFs with 4096 basis functions because the nearest-neighbor calculations for that graph proved too expensive). We systematically varied α (the gradient descent term) to optimize performance for each combination of basis type and order. Table 4(a) shows the resulting α values.[3]

The PVFs were built using the normalized Laplacian: $L = D^{-1/2}(D - W)D^{-1/2}$, scaling the resulting eigenvectors to the range $[-1, 1]$. We also rescaled the Acrobot state variables by $[1, 1, 0.05, 0.03]$ for local distance calculations. Random walks did not suffice to collect samples that adequately covered the state space, so we biased the sample collection to only keep examples where the random walk actually reached the target within 800 steps. We kept 50 of these episodes—approximately 3200 samples. We subsampled these initial samples to graphs of 1200 or 1400 points, created using the nearest 30 neighbors. We did not optimize these settings as well as might have been possible, and thus the PVF averages are not as good as they could have been. However, our intention was to consider the performance of each method as generally used, and a full parameter exploration was beyond the scope of this work.

We ran 20 episodes averaged over 100 runs for each type of learner. The results, shown in Figure 4, demonstrate that the Fourier basis learners outperformed all other types of learning agents for all sizes of function approximators. In particular, the Fourier basis performs better initially than the polynomial basis (which generalizes broadly) and converges to a better policy than the RBFs (which generalize locally). It also performs slightly better than PVFs, even though the Fourier basis is a fixed, rather than learned, basis. However, the value function for Acrobot does not contain any discontinuities, and is therefore not the type of problem that PVFs are designed to solve. In the next section, we consider a domain with a discontinuity in the value function.

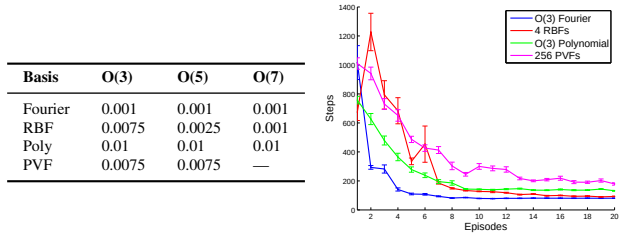
The Discontinuous Room. In the Discontinuous Room, shown in Figure 5(a), an agent in a 10×6 room must reach a target; the direct path to the target is blocked by a wall, which the agent must go around. The agent has four actions which move it 0.5 units in each direction. This is a simple continuous domain with a large discontinuity, which we use to compare the Fourier basis to PVFs, which were specifically designed to handle discontinuities by modeling the connectivity of the state space. In order to make a reliable comparison, agents using PVFs were given a perfect adjacency matrix containing every legal state and transition, thus avoiding sampling issues and removing the need for an out-of-sample extension method.

Figure 5(b) shows learning curves for the Discontinuous Room using Sarsa(λ) ($\gamma = 0.9$, $\lambda = 0.9$, $\epsilon = 0$) using Fourier bases of order 5 and 7 ($\alpha = 0.001$ in both cases) defined over the two state variables, and agents using 36 and 64 ($\alpha = 0.025$ in both cases) PVFs. The Fourier basis outperforms PVFs. Figures 5(c) and 5(d) show example value

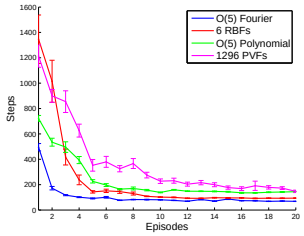
[3]Note that α is a gradient descent coefficient; during learning it is multiplied by each basis function to modify that function's weight. Hence, its value is not comparable across different bases.

Basis	O(3)	O(5)	O(7)
Fourier	0.001	0.001	0.001
RBF	0.0075	0.0025	0.001
Poly	0.01	0.01	0.01
PVF	0.0075	0.0075	—

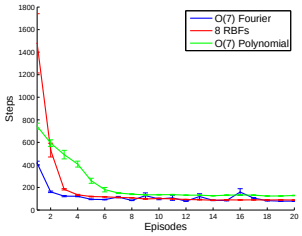
(a)



(b)

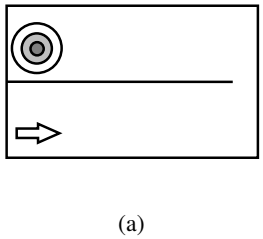


(c)

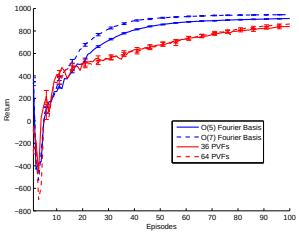


(d)

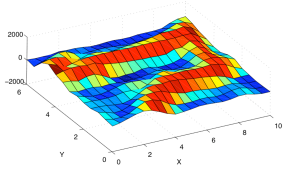
Figure 4: α values used for the Swing-Up Acrobot; learning curves for agents using (b) order 3 (c) order 5 and (d) order 7 Fourier bases, and RBFs and PVFs with corresponding number of basis functions. Error bars are standard error.



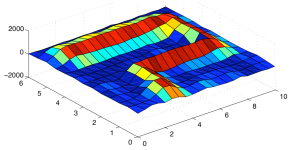
(a)



(b)



(c)



(d)

Figure 5: The Discontinuous Room (a), where an agent must move from the lower left corner to the upper left corner by skirting around a wall through the middle of the room. Learning curves (b) for agents using $O(5)$ and $O(7)$ Fourier bases and agents using 36 and 64 PVFs. Error bars are standard error. Value functions from agents using an $O(7)$ Fourier basis (c) and 64 PVFs (d).

functions for an agent using an order 7 Fourier basis and a corresponding agent using 64 PVFs. The PVF value function is clearly a better approximation, very precisely modeling the discontinuity around the wall. In contrast, the Fourier basis value function is noisier and does not model the discontinuity as cleanly. However, the results in Figure 5(b) suggest that this does not significantly impact the quality of

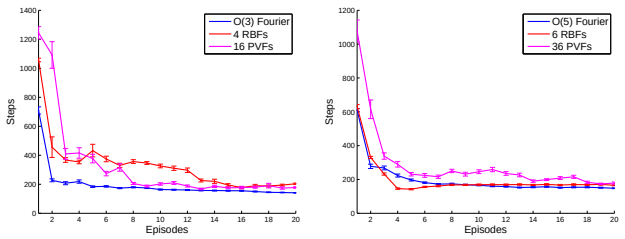
the resulting policy; it appears that the Fourier basis agent has learned to avoid the discontinuity. This suggests that the Fourier basis is sufficient for problems with a small number of discontinuities, and that the extra complexity (in samples and computation) of PVFs only becomes really necessary for more complex problems.

Mountain Car. The Mountain Car (Sutton and Barto, 1998) is an underpowered car stuck in a valley; to escape, it must first accelerate up the back of the valley, and then use the resulting momentum to drive up the other side. This induces a steep discontinuity in the value function which makes learning difficult for bases with global support.

We employed Sarsa(λ) ($\gamma = 1.0$, $\lambda = 0.9$, $\epsilon = 0$) with Fourier bases of orders 3 and 5, and RBFs and PVFs of equivalent sizes (we were unable to learn with the polynomial basis). The α values used are in Table 1.

Basis	O(3)	O(5)
Fourier	0.001	0.001
RBF	0.025	0.025
PVF	0.01	0.025

Table 1: α values used for Mountain Car.



(a) (b)

Figure 6: Mountain Car learning curves for agents using (a) order 3 (b) order 5 Fourier bases, and RBFs and PVFs with corresponding number of basis functions. Error bars are standard error.

The results (shown in Figure 6) indicate that for low orders, the Fourier basis outperforms RBFs and PVFs. For higher orders the Fourier basis initially performs worse (because it does not model the discontinuity well) but converges to a better solution.

Discussion

Our results show that the Fourier basis provides a reliable basis for value function approximation. We expect that for many problems, the Fourier basis will be sufficient to learn a good value function without requiring transition samples or an expensive or complex method to create a basis. Additionally, the ability to include prior knowledge about variable interaction provides a simple and useful tool for inducing structure in the value function.

Another vein of research has examined multi-scale basis function construction, which create a hierarchy of basis functions at different levels of resolution. One approach employs multigrid methods to construct basis functions at multiple levels of resolution (Ziv and Shimkin, 2005). Diffusion wavelets (Mahadevan and Maggioni, 2006) are another approach that compactly represents dyadic powers of the transition matrix at each level of the hierarchy. We examined the utility of a simple global fixed basis that does not require sampling to construct; a similar examination of wavelet basis functions may provide interesting results.

The one area in which we find that the Fourier basis has difficulty is representing flat areas in value functions; moreover, for high order Fourier basis approximations, sudden discontinuities may induce "ringing" around the point of discontinuity, known as *Gibbs phenomenon* (Gibbs, 1898). This may result in policy difficulties near the discontinuity.

Another challenge posed by the Fourier basis (and all fixed basis function approximators) is how to select basis functions when a full order approximation of reasonable size is too large to be useful. PVFs, by contrast, scale with the size of the sample graph (and thus could potentially take advantage of the underlying complexity of the manifold). One potential solution is the use of feature selection (Kolter and Ng, 2009; Johns, Painter-Wakefield, and Parr, 2010) to obtain a good set of basis functions for learning. The Fourier basis is particularly well suited to feature selection because it provides an easy way to specify a fixed yet complete (up to a given order) set of basis functions, where the important relationships between state variables may be easily interpreted by a human through the $\mathbf{c}$ vectors corresponding to selected basis functions. For example, if basis functions corresponding to high-order interactions between two variables are consistently selected it would be reasonable to infer that that interaction is important to the dynamics of the domain.

Our experiences with the Fourier basis show that this simple and easy to use basis reliably performs well on a range of problems. As such, although it may not perform well for domains with several significant discontinuities, its simplicity and reliability suggest that linear function approximation using the Fourier basis should be the first approach tried when RL is applied to a new problem.

Summary

We have described the Fourier basis, a linear function approximation scheme using the terms of the Fourier series as features. Our experimental results show that this simple and easy to use basis performs well compared to two popular fixed bases and a learned set of basis functions in three continuous RL benchmarks, and is competitive with a popular learned basis.

Acknowledgements

SO was supported in part by the National Science Foundation under grants NSF IIS-0534999 and NSF IIS-0803288. GDK was supported in part by the AFOSR under grant AOARD-104135 and the Singapore Ministry of Education under a grant to the Singapore-MIT International Design Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, the AFOSR, or the Singapore Ministry of Education.

Appendix: Scaling α

When performing batch gradient descent on an error function, Armijo (1966) and Plagianakos, Vrahatis, and Magoulas (1999) suggested that the learning rate should be inversely proportional to a Lipschitz constant of the update term $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$, which is defined in Euclidean space as an L satisfying $\left|\left|\frac{\partial E(\mathbf{w}_1)}{\partial \mathbf{w}} - \frac{\partial E(\mathbf{w}_2)}{\partial \mathbf{w}}\right|\right|_2 \leq L||\mathbf{w}_1 - \mathbf{w}_2||_2$, for all $\mathbf{w}_1$ and $\mathbf{w}_2$, where E is the error function being minimized (e.g. Bellman error for value estimate with weights $\mathbf{w}$). For any of the error functions in C^2 that are typically used, this implies that the learning rate should be inversely proportional to the maximum magnitude of the second derivative of the error function with respect to the weights: $\alpha \propto \frac{1}{L} \leq \min_{\mathbf{w}} \left|\left|\frac{\partial E(\mathbf{w})}{\partial^2 \mathbf{w}}\right|\right|_2^{-1}$.

However, Armijo's method for selecting a learning rate only applies to batch gradient descent with a single learning rate, and therefore only accounts for the morphology of the error surface as a function of the weights, and not states or individual features. Whereas the gradient in batch gradient descent is only a function of $\mathbf{w}$, when learning a value function with stochastic gradient descent, it also depends on the current state $\mathbf{x}$. Thus, while Armijo's method acts to limit the difference during batch gradient descent in the update term $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$ between consecutive updates, we propose that the difference in the stochastic gradient descent update term $\frac{\partial E(\mathbf{w},\mathbf{x})}{\partial \mathbf{w}}$ between consecutive updates should be limited.

We therefore modify Armijo's method to the stochastic gradient descent case by setting each terms' learning rate inversely proportional to a Lipschitz constant of $\frac{\partial E(\mathbf{x},\mathbf{w})}{\partial w_i}$, which is defined in Euclidean space as a K_i satisfying $\left|\left|\frac{\partial E(\mathbf{y}_1)}{\partial w_i} - \frac{\partial E(\mathbf{y}_2)}{\partial w_i}\right|\right|_2 \leq K_i \left|\left|\mathbf{y}_1 - \mathbf{y}_2\right|\right|_2$, for all $\mathbf{y}_1$ and $\mathbf{y}_2$ where $\mathbf{y} = [\mathbf{x}\ \mathbf{w}]$ and $E(\mathbf{y}) = E(\mathbf{x}, \mathbf{w})$ is the error in state $\mathbf{x}$ with weighting $\mathbf{w}$. To derive estimates of learning rate proportions, we assume we are performing stochastic gradient descent with the error function $E(\mathbf{x}, \mathbf{w}) = \frac{1}{2}\left(V(\mathbf{x}) - \bar{V}(\mathbf{x})\right)^2$.

We assume that the learning rate α_1 for a term with unit coefficient vector, $\left|\left|\mathbf{c}^1\right|\right|_2 = 1$, is provided, and then scale the learning rate for each other term as

$$\alpha_i = \alpha_1 \frac{K_1}{K_i}, \tag{5}$$

where K_i is a Lipschitz constant for the term with coefficient vector $\mathbf{c}^i$ and weight w_i:

$$K_i \geq \max_{\mathbf{y}} \left|\left| \frac{\partial}{\partial \mathbf{y}} \frac{\partial E(\mathbf{y})}{\partial w_i} \right|\right|_2. \tag{6}$$

Our goal is to find the smallest value for K_i that satisfies Equation 6 while producing a simple conclusion. Because $\left|\left|\frac{\partial}{\partial \mathbf{x}} \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial w_i}\right|\right|_2 \leq \left|\left|\pi \mathbf{c}^i \left(V(\mathbf{x}) - \mathbf{w} \cdot \phi(\mathbf{x})\right) + \frac{\partial}{\partial x}\left(V(\mathbf{x}) - \mathbf{w} \cdot \phi(\mathbf{x})\right)\right|\right|_2$ and $\left|\left|\frac{\partial}{\partial \mathbf{w}} \frac{\partial E(\mathbf{x},\mathbf{w})}{\partial w_i}\right|\right|_2 \leq m^{\frac{1}{2}} \leq \left|\left|\mathbf{c}^i\right|\right|_2 m^{\frac{1}{2}}$ for m basis functions, we can select $K_i = \left|\left|\mathbf{c}^i\right|\right|_2 \left[m^{\frac{1}{2}} + \max_{\mathbf{x},\mathbf{w}} \left(||\pi\left(V(\mathbf{x}) - \mathbf{w} \cdot \phi(\mathbf{x})\right)||_2 + \left|\left|\frac{\partial}{\partial x}\left(V(\mathbf{x}) - \mathbf{w} \cdot \phi(\mathbf{x})\right)\right|\right|_2\right)\right]$, which satisfies Equation 6 for all $i \neq 0$. Substituting into Equation 5, we conclude that the i^{th} term's learning rate should be $\alpha_i = \alpha_1 / \left|\left|\mathbf{c}^i\right|\right|_2$. To avoid division by zero for $\mathbf{c}^0 = 0$, we select $\alpha_0 = \alpha_1$.

References

Aleida, L.; Langlois, T.; Amaral, J.; and A.Plakhov. 1998. Parameter adaptation in stochastic optimization. In Saad, D., ed., *Online Learning in Neural Networks*. Cambridge, MA: Cambridge University Press. 111–134.

Armijo, L. 1966. Minimization of functions having Lipschitz continuous first partial derivatives. In *Pacific Journal of Mathematics*.

Gibbs, J. 1898. Fourier series. *Nature* 59(200).

Johns, J.; Painter-Wakefield, C.; and Parr, R. 2010. Linear complementarity for regularized policy evaluation and improvement. In *Advances in Neural Information Processing Systems 23*.

Kolter, J., and Ng, A. 2007. Learning omnidirectional path following using dimensionality reduction. In *Proceedings of Robotics: Science and Systems*.

Kolter, J., and Ng, A. 2009. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th International Conference on Machine Learning*, 521–528.

Konidaris, G., and Osentoski, S. 2008. Value function approximation in reinforcement learning using the Fourier basis. Technical Report UM-CS-2008-19, Department of Computer Science, University of Massachusetts, Amherst.

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

Mahadevan, S., and Maggioni, M. 2006. Value function approximation using diffusion wavelets and Laplacian eigenfunctions. In *Neural Information Processing Systems*. MIT Press.

Mahadevan, S., and Maggioni, M. 2007. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research* 8:2169–2231.

Mahadevan, S.; Maggioni, M.; Ferguson, K.; and Osentoski, S. 2006. Learning representation and control in continuous Markov decision processes. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*.

Plagianakos, V.; Vrahatis, M.; and Magoulas, G. 1999. Nonmonotone methods for backpropagation training with adaptive learning rate. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, 1762–1767.

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

Tanner, B., and White, A. 2009. RL-Glue : Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research* 10:2133–2136.

Ziv, O., and Shimkin, N. 2005. Multigrid methods for policy evaluation and reinforcement learning. In *International Symposium on Intelligent Control*, 1391–1396.