Adaptive Online Value Function Approximation with Wavelets

Michael Beukman
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
michael.beukman1@students.wits.ac.za

Michael Mitchley
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
barcoded@gmail.com

Dean Wookey
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
wookey.dean@gmail.com

Steven James
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
steven.james@wits.ac.za

George Konidaris
Department of Computer Science
Brown University
Providence RI, 02912
gdk@cs.brown.edu

Abstract

Using function approximation to represent a value function is necessary for continuous and high-dimensional state spaces. Linear function approximation has desirable theoretical guarantees and often requires less compute and samples than neural networks, but most approaches suffer from an exponential growth in the number of functions as the dimensionality of the state space increases. In this work, we introduce the wavelet basis for reinforcement learning. Wavelets can effectively be used as a fixed basis and additionally provide the ability to adaptively refine the basis set as learning progresses, making it feasible to start with a minimal basis set. This adaptive method can either increase the granularity of the approximation at a point in state space, or add in interactions between different dimensions as necessary. We prove that wavelets are both necessary and sufficient if we wish to construct a function approximator that can be adaptively refined without loss of precision. We further demonstrate that a fixed wavelet basis set performs comparably against the high-performing Fourier basis on Mountain Car and Acrobot, and that the adaptive methods provide a convenient approach to addressing an oversized initial basis set, while demonstrating performance comparable to, or greater than, the fixed wavelet basis. To aid in reproducibility, we publicly release our source code.[1]

Keywords: reinforcement learning, value function, wavelets, linear function approximation

Acknowledgements

Computations were performed using High Performance Computing infrastructure provided by the Mathematical Sciences Support unit at the University of the Witwatersrand.

[1] https://github.com/Michael-Beukman/WaveletRL

1 Introduction

Representing a value function in reinforcement learning (RL) in continuous state spaces requires function approximation. One approach is non-linear function approximation, where a neural network is trained to learn useful features from raw observations. However, this class of methods requires many samples, large amounts of computation [1], and does not possess theoretical or convergence guarantees [2]. Another common scheme is *linear function approximation*, where the value function is approximated by a weighted sum of non-learnable basis functions. This results in simple algorithms and an error surface convex in the weights, but still allows for the representation of complex value functions because the basis functions themselves can be arbitrarily complex. The obvious question is then: which basis functions should one use?

Several fixed basis schemes have been introduced (e.g., [3–5]), all with the inherent disadvantage of a combinatorial explosion that makes them unsuited to high-dimensional domains. Consequently, research has focused on either constructing basis functions from data in both discrete and continuous state spaces, or selecting an appropriate set from a fixed dictionary of candidate basis functions with the aim of producing basis function sets that are subexponential in the number of dimensions. A third approach is *adaptive methods* (e.g., [6–9]), which are a hybrid of the above. Adaptive methods create new basis functions that complement or replace an existing set of basis functions on which learning is performed. They have the advantage in that they are online, which reduces the sample and computational complexity, and do not require a large dictionary of candidate functions, since they add representation complexity incrementally.

We extend existing adaptive methods to basis functions based on *wavelets*, which are able to approximate functions at various scales and locations, and which can be refined to build a more accurate representation where such detail is necessary. We further prove that wavelets are both necessary and sufficient for function splitting approaches. We present an algorithm for value function approximation that begins with a minimal set of basis functions, and only adds interactions between different dimensions as required. We test our approach on Mountain Car and Acrobot and our results show that a fixed wavelet basis is competitive with the high-performing Fourier basis [4]. Furthermore, the adaptive techniques perform comparably to the fixed basis, while not starting out with a complete basis set.

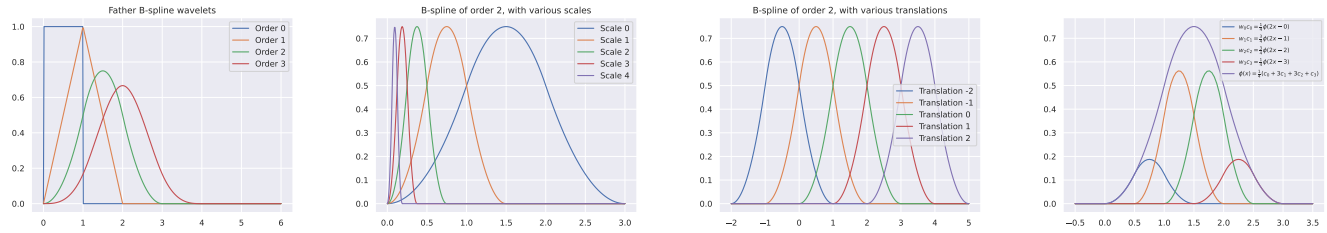
2 Value Function Approximation

The reinforcement learning problem in continuous domains is typically modelled as a Markov Decision Process and described by a tuple $(S; A; P; R; \ )$, where $S \quad \mathbb{R}^d$ is a d-dimensional state space, A is a set of actions, $P(s^0 | s; a)$ describes the probability of transitioning to state s^0 after having performed action a in state s, $R(s; a)$ describes the reward received for executing such a transition, and $\ $ is the discount factor. The agent is required to learn a policy $\ $ mapping states to actions that maximises the return (discounted future sum of rewards) at time t: $G_t = \sum_{i=0}^{1} {}^i R(s_{t+i}; a_{t+i})$ [2]. Given a policy $\ $, RL algorithms often estimate a *value function*: $V \ (s) = \mathbb{E}\left[\sum_{i=0}^{1} {}^i R(s_i; a_i) | s_0 = s \right]$, where action a is selected according to policy $\ $. Linear value function approximation represents $V \ $ as weighted sum of n basis function $\ $: $V \ (s) \quad \mathbf{w} \quad = \sum_{i=1}^{n} w_i \ _i(s)$. This approximation is linear in the components of the parameter (or weight) vector, $\mathbf{w}$, which results in simple update rules and a quadratic error surface.

Basis Functions In function approximation, a *basis* is a set of orthonormal functions spanning a function space, such that anything within that function space can be exactly reconstructed using a unique weighted sum of the basis functions. Typically, we are interested in orthonormal bases for $L_2(\mathbb{R})$, the space of all finite square-integrable functions. In Euclidean spaces, $f(x) \ 2 \ L_2(\mathbb{R})$ implies $\int_{1}^{1} f^2(x) dx$ is finite, a property satisfied by all value functions with finite return. Formally, if a basis is overcomplete (i.e. not all functions are orthogonal), then it is known as a *frame*.

A simple choice of basis function is *tile coding* [5], where the state space is discretised into tile basis functions that evaluate to 1 within a fixed region, and 0 elsewhere. Although tile coding forms a basis, it restricts the value function to be piecewise constant. Another approach is the *polynomial basis* [10], which sets $_i(s) = \prod_{j=1}^{d} s_j^{c_{i,j}}$, where each $c_{i;j} \ 2 \ [0; : : : ; n]$ and n denotes the order of the basis. While orthonormal polynomials exist, simple ones are not, and may lead to redundancies in their representations. A more common scheme is *radial basis functions* (RBFs) [3], where each function is a Gaussian with a given mean and variance. RBFs have local support and are therefore well-suited to representing value functions with local discontinuities. The *Fourier basis* [4] uses Fourier series terms as basis functions, setting $_i(s) = \cos(\ \mathbf{c}_i \ s)$ where for order n, $\mathbf{c}_i = [c_1; : : : ; c_d]$ is a vector of coefficients between 0 and n. The Fourier basis requires choosing just a single parameter (the order) and in practice outperforms RBFs and polynomials on some common low-dimensional benchmarks [4].

The main shortcoming of these fixed-basis approaches is that the number of basis functions grows exponentially with the dimension of the state space. This has led to feature selection approaches that use a fixed basis as a feature dictionary, but only select a small subset of terms to compactly represent the value function. Ideally, such methods would allow the function approximator to represent extra detail where necessary and handle local discontinuities. However, the above basis function schemes are poorly suited to this: tile coding requires careful discretisation and can only represent piecewise-constant functions, RBFs lack an obvious way of setting their centres and variances, and the Fourier and



(a) Different orders (b) Scaling (c) Translating (d) Refineability

Figure 1: B-Spline wavelets of different (a) orders, (b) scales and (c) translations. In (d), the original function (purple) can be represented as a weighted sum of its "children" functions. For clarity, these functions are not normalised.

polynomial bases produce functions with global support. We therefore require a new basis scheme suitable for general function approximation that allows us to add spatially local basis functions to incrementally add detail to the value function representation.

3 Wavelets

Through the use of a *Fourier transform*, we can represent any periodic function as an integral of sines and cosines with varying frequencies. However, since sines and cosines have global support, the Fourier transform becomes cumbersome when representing *transient* or local phenomena [11]. To deal with this issue, we can instead consider *wavelets*. Wavelets are simply functions $: \mathbb{R} \to \mathbb{R}$, some families of which are compactly supported [12] (nonzero in a finite interval) allowing us to model local phenomena. There are many types of wavelets but we focus on B-Spline father wavelets here, the first three orders of which are given by:

$$
{}^{0}(x) = \begin{cases} 1 & 0 \quad x \quad 1 \\ 0 & \text{otherwise} \end{cases}
\qquad
{}^{1}(x) = \begin{cases} x & 0 \quad x \quad 1 \\ 2 \quad x & 1 \quad x \quad 2 \\ 0 & \text{otherwise} \end{cases}
\qquad
{}^{2}(x) = \begin{cases} 0{:}5x^2 & 0 \quad x \quad 1 \\ 0{:}75 \quad (x \quad 1{:}5)^2 & 1 \quad x \quad 2 \\ 0{:}5(x \quad 3)^3 & 2 \quad x \quad 3 \\ 0 & \text{otherwise} \end{cases}
\tag{1}
$$

In practice, these functions are normalised to satisfy $\int_{1}^{1} (\ (x))^2 dx = 1$. We note that the zero$^{\text{th}}$ order B-Spline function is also referred to as a Haar wavelet [13], and can produce functions equivalent to disjoint tile coding. We can scale (dilate) an arbitrary wavelet of order n by multiplying the input $x \in \mathbb{R}$ by 2^j to obtain ${}^{n}_{j}(x) = {}^{n}(2^j x)$. We can further translate it by $k \in \mathbb{Z}$ to obtain ${}^{n}_{j;k}(x) = {}^{n}(2^j x \quad k)$. Some examples are shown in Figure 1. One appealing property of wavelets is that they are *refinable*, meaning that they can be replaced with a weighed sum of smaller copies of themselves at regular intervals. Concretely, as shown in Figure 1d, they satisfy the *refinability equation*:

$$
(x) = \sum_{k} w_k \ (mx \quad k) \text{ with } m > 1; k \in \mathbb{Z} \tag{2}
$$

We now prove that wavelets are both necessary and sufficient as a basis that obeys this equation.

Theorem 1. *If one wishes to split a basis function* $(x) \in L_2(\mathbb{R})$ *(that is, the basis function is finite square-integrable) into a finite number of smaller copies of itself such that the value function remains unchanged, using Equation 2 where* $m > 1$ *and* $k \in \mathbb{Z}$*, with a finite number of* w_k *nonzero, it is necessary and sufficient to use a compactly supported father wavelet as* .

Proof. The above conditions define a father wavelet function [14]. Further, Equation 2 is obeyed by all father wavelet functions with refinement mask m [12]. If a father wavelet has compact support, it will have a finite dilation series. □

3.1 Wavelets as a basis for linear function approximation in reinforcement learning

To construct the wavelet basis for RL, given a state space $S \quad [0; 1]^d$, we choose a specific order n and scale j. For each dimension of the state space, we create wavelets with scale j, order n and translations $n \quad k < 2^j$. The full basis then consists of all combinations (products) of d of these functions, such that each term has a unique dimension. This is a fixed basis, providing a drop-in replacement for the methods described above. As an example, with $d = 2; n = 1; j = 0$, we have state $[s_1; s_2]$ and the atomic functions are $f \ (s_1 \quad 0); \ (s_1 + 1); \ (s_2 \quad 0); \ (s_2 + 1)g$. All pairwise products with distinct dimensions yield $= [\ (s_1 \quad 0) \ (s_2 \quad 0); \ (s_1 \quad 0) \ (s_2 + 1); \ (s_1 + 1) \ (s_2 \quad 0); \ (s_1 + 1) \ (s_2 + 1)]$.

Adaptive Methods: This still has an exponential number of terms, however. To remedy this, we can start with a minimal basis set and add extra resolution *as and when necessary*. This can be done using two atomic operations: *splitting* and

combining. The former takes a feature and replaces it with its children, redistributing its weight among the children, such that the value function remains unchanged. This allows each child's weight to be updated individually, adding more representational capacity. In the above example, if we were to refine $_1$ in dimension 1, we obtain the children of (s_1): $(2s_1)$; $(2s_1 \quad 1)$; $(2s_1 \quad 2)$ and multiply each child with (s_2) to obtain 3 new functions, which would replace $_1$. The second operation, *combining*, adds in products between different dimensions dynamically, instead of starting with a full combination. Intuitively, this allows us to start with a decoupled basis set (with $(n + 2^j)d$ terms as opposed to $(n + 2^j)^d$ for a fully coupled basis), where interactions between different state variables are ignored, and only added in when doing so would improve our approximation of the value function. For example, the initial basis set would simply be $= [(s_1); (s_1 + 1); (s_2); (s_2 + 1)]$, and conjunctions could be subsequently added.

However, this raises a new question: *how do we choose which functions to split and which to combine?* One solution is to estimate the usefulness or *relevance* of a candidate function as we learn the value function. We first define $H(; E) = \frac{T}{T} \frac{1}{jj \quad jj(1 \quad)} \sum_{t=0}^{T} {}^{(T \quad t)} E(s_t) \quad (s_t)$ where T is the sample count of (the number of times that was nonzero for a given state), $jj \quad jj$ is the size of the domain in which is nonzero and is a hyperparameter which controls how much weight in the moving average is put on recent samples. We then extend the relevance measure of Geramifard et al. [8] to obtain the *relevance* $( ) = H(; E(s_t) = (s_t))$ and the *observed error* $O( ) = H(; E(s_t) = j \ (s_t)j)$, where $_t$ is the *TD error* $_t = R(s_t; a_t) + V(s_{t+1}) \quad V(s_t)$. These quantities are estimates of the inner products $h \ ; \ i$ and $h \ ; j \ ji$, respectively. Intuitively, functions with high relevances will be frequently nonzero in the presence of error and will thus reduce this error when added to the basis set. Finally, we define $C( ) = O( ) \quad ( )$ for convenience.

Using the above ideas of *splitting*, *combining* and *relevance*, we first present AWR (Algorithm 1), in which we start with a full basis set at some scale and adaptively increase the scale for functions that have the highest C above some tolerance. This is similar to the method proposed by Li and Zhu [9] (who used Haar wavelets, although the theoretical analysis is more generally applicable), but we split based on the function relevance instead of using the coefficient magnitudes. The next algorithm is IBFDD (Algorithm 2), which starts with a decoupled basis set and adds in interactions (i.e. products) of different basis functions as the method progresses. Finally, we combine these two to obtain the Multiscale Adaptive Wavelet Basis (MAWB), which simply starts with a decoupled basis set, and alternates between IBFDD and AWR as learning progresses—either increasing the detail in a local region or adding in useful interactions.

Algorithm 1 AWR: Adaptive Wavelet Refinement

Require: Basis set $\mathbf{F}$, s, δ_t, $\rho(\phi)$, $T(\phi)$, splitting tolerance τ_s.
  for all Functions ϕ activated by state (s, a) **do**
    Update $\rho(\phi)$, $O(\phi)$, $T(\phi)$ & $\rho(\phi_c), T(\phi_c)$ $\forall$ children ϕ_c of ϕ.
  end for
  if $C(\phi) \geq \tau_s$ and $C(\phi) = \max_k(C(\phi_k))$ **then**
    Replace ϕ with its children in dimension d, $\phi^d_{j,k}$, where d maximises the average relevance of the children of ϕ.
  end if

Algorithm 2 IBFDD: Incremental Basis Function Dependency Discovery

Require: Basis set $\mathbf{F}$, s, δ_t, $\rho(\phi)$, $T(\phi)$, combination tolerance τ_c.
  for all $\phi_f = \phi_g \phi_h$ s.t. $\phi_g, \phi_h \in \mathbf{F}, \phi_f \notin \mathbf{F}$ and $\phi_f(s) \neq 0$ **do**
    Update $\rho(\phi_f), T(\phi_f)$
  end for
  if $|\rho(\phi_f)| \geq \tau_c$ and $|\rho(\phi_f)| = \max_k |\rho(\phi_k)|$ **then**
    $\mathbf{F} \leftarrow \mathbf{F} \cup \{\phi_f\}$
  end if

3.2 Preliminary Experiments

In Figure 2 we demonstrate the performance of the wavelet basis on two tasks using Sarsa( ). We see that, for Mountain Car, the fixed B-spline basis with 36 terms in total is competitive with the Fourier basis with 36 terms. For Acrobot, most methods do well, and again, the B-Spline basis is competitive with the Fourier basis. The adaptive methods generally perform comparably to the fixed ones, while not needing to start with an exponentially sized basis set. We do note, however, that using a decoupled basis (which is equivalent to MAWB with $_c; \ _s = 1$) often outperformed a coupled one, motivating the need to investigate more complex environments. Finally, we show example value functions in Figure 3.

4 Conclusion

We introduced a linear function approximation scheme whose features are wavelet functions. We proved that the use of wavelets is both necessary and sufficient to obtain a refinable basis of $L_2(\mathsf{R})$ that can perform multi-scale function approximation. Preliminary experimental results demonstrate that wavelets are competitive with other fixed-basis function approximation schemes.

References

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.