Incremental Basis Construction from Temporal Difference Error

Yi Sun YI@IDSIA.CH
Faustino Gomez TINO@IDSIA.CH
Mark Ring MARK@IDSIA.CH
Jürgen Schmidhuber JUERGEN@IDSIA.CH
IDSIA, USI & SUPSI, Switzerland

Abstract

In many reinforcement learning (RL) systems, the value function is approximated as a linear combination of a fixed set of basis functions. Performance can be improved by adding to this set. Previous approaches construct a series of basis functions that in sufficient number can eventually represent the value function. In contrast, we show that there is a single, *ideal* basis function, which can directly represent the value function. Its addition to the set immediately reduces the error to zero—without changing existing weights. Moreover, this ideal basis function is simply the value function that results from replacing the MDP's reward function with its Bellman error. This result suggests a novel method for improving value-function estimation: a primary reinforcement learner estimates its value function using its present basis functions; it then sends its TD error to a secondary learner, which interprets that error as a reward function and estimates the corresponding value function; the resulting value function then becomes the primary learner's new basis function. We present both batch and online versions in combination with incremental basis projection, and demonstrate that the performance is superior to existing methods, especially in the case of large discount factors.

1. Introduction

The state space of most real-world reinforcement learning (RL) problems is too large to permit a tabular representation of the value function, and therefore the value function must be approximated. Linear function approximators (LFAs) are popular for this purpose due to their theoretical simplicity and low computational complexity (Sutton and Barto, 1998; Boyan, 2002; Lagoudakis and Parr, 2003; Sutton et al., 2009; Maei and Sutton, 2010). Learning in LFAs hinges on two intertwined subproblems: *weight optimization*, which aims to effectively compute the weights for a given set of basis functions, and *basis discovery*, which is concerned with finding good basis functions.

The last two decades have seen significant progress addressing weight optimization, particularly in the context of temporal-difference (TD) methods (Sutton, 1988). Second-order TD-methods such as residual gradient (Baird, 1995) and LSTD (Bradtke et al., 1996; Boyan, 2002; Geramifard et al., 2006) have become standard tools for solving reinforcement learning problems with thousands of basis functions. More recently, second-order, online TD-methods with complexity linear in the number of basis functions have successfully solved large-scale problems with tens of thousands of basis functions (Sutton et al., 2008; 2009).

While the weight optimization problem has largely been solved, the basis discovery problem remains open. A significant fraction of the existing approaches make use of the so-called Bellman error basis functions (BEBFs; Wu and Givan (2005); Keller et al. (2006); Parr et al. (2007); Mahadevan and Liu (2010)), which capture the intuition that the "Bellman error, loosely speaking, point[s] towards the optimal value function" (Parr et al., 2007). Because a sequence of normalized BEBFs form an orthonomal basis of the space in which the value function resides, any value function can be exactly represented given a sufficient number of BEBFs (Parr et al., 2007; Mahadevan and Liu, 2010). Some approaches that do not rely on BEBFs include the construction of reward-insensitive "proto value" basis functions from the expansion of the Graph Laplacian (Mahadevan et al., 2006), basis selection under sparsity constraints (Kolter and Ng, 2009), and basis projection from predictive compression of observations

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

(Boots and Gordon, 2010).

In this paper, we pursue an entirely different idea for basis-function generation. Instead of constructing a *sequence* of basis functions that, in sufficient number, can eventually represent the value function, we aim to do so with a single new basis function. Our approach is based on the simple insight that, for a given MDP and set of basis functions, there is an 'ideal' basis function, which can theoretically reduce the error to zero immediately. This ideal basis function is simply *the value function that results from replacing the MDP's reward function with its Bellman error*. We refer to this basis function as the 'Value Function of the Bellman Error,' and denote it V-BEBF. V-BEBF allows the value function to be represented exactly, without affecting the optimal weight values for the current basis functions.

This finding directly transforms basis discovery into a reinforcement learning problem, suggesting a novel approach for incrementally expanding the set of basis functions: a primary reinforcement learner estimates its value function over a set of basis functions; it then uses its TD-error as the reward function for a second reinforcement learner on the same MDP; the resulting value function is returned to the primary learner as a new basis function. We present both batch and online versions of this approach using the *linear basis projection* framework suggested by Boots and Gordon (2010) and Ghavamzadeh et al. (2010). We then demonstrate the effectiveness of both approaches experimentally.

The formulation of V-BEBF is detailed in Section 3, after some necessary background material presented in Section 2. Incremental basis projection with V-BEBF is described in Section 4, and empirical studies are presented in Section 5.

2. Background

Without loss of generality, we describe V-BEBF with respect to value-function estimation for a Markov Decision Process (MDP) with fixed policy, a finite state space $\mathcal{S} = \{1, \ldots, S\}$, an expected reward function[1] r, and a discount factor $\gamma \in [0, 1)$. Let P be the transition matrix of the Markov chain resulting from marginalizing out the actions, then the value function v solves the Bellman equation

$$v = r + \gamma P v.$$

which can be conveniently written as

$$v = L^{-} r, \tag{1}$$

where[2] $L = I - \gamma P$.

When S is too large for v to be maintained in tabular form, it is common to approximate v via a linear combination of *basis functions* $\Phi = [\phi_1, \cdots, \phi_N]$ such that

$$v \simeq \sum_{n=1}^{N} \theta_n \phi_n = \Phi\theta,$$

where $N \ll S$ and $\theta = [\theta_1, \ldots, \theta_N]^{\top}$ are the corresponding weights. The *Bellman error* is given by

$$\varepsilon = r + \gamma P \Phi\theta - \Phi\theta = r - L\Phi\theta = L\left(v - \Phi\theta\right).$$

Clearly, $\varepsilon = 0$ if and only if $v = \Phi\theta$.

When the basis functions are fixed, the weights can be obtained using second-order TD methods, which minimize quadratic objective functions of the form

$$J = \left(L^{-}\varepsilon\right)^{\top} K \left(L^{-}\varepsilon\right) = \left(v - \Phi\theta\right)^{\top} K \left(v - \Phi\theta\right). \tag{2}$$

The solution is computed either explicitly, or through (stochastic or batch) gradient descent. The positive semi-definite matrix K differs from method to method, and examples include $K_{rg} = L^{\top} D L$ for residual gradient (Baird, 1995), $K_{gtd} = (\Phi D L)^{\top} (\Phi D L)$ for GTD(0) (Sutton et al., 2008), and $K_{lstd} = L^{\top} \Pi_D D \Pi_D L$ for LSTD (Bradtke et al., 1996) and TDC (Sutton et al., 2009). Here D is the diagonal matrix whose diagonal is the sample distribution of the states (which coincides with the stationary distribution of P if the learning is on-policy and the Markov chain is ergodic), and $\Pi_A = \Phi\left(\Phi^{\top} A \Phi\right)^{-} \Phi^{\top} A$ is the projection operator with respect to the (semi) inner product defined by A as $\langle v_1, v_2 \rangle_A = v_1^{\top} A v_2$, for a given Φ, and an arbitrary positive semi-definite matrix A.

The negative gradient of the objective function J with respect to θ is

$$-\frac{1}{2} \nabla_{\theta} J = \Phi^{\top} K L^{-} \varepsilon,$$

and the weights minimizing J, to which the second order TD methods converge, are given by

$$\hat{\theta} = \left(\Phi^{\top} K \Phi\right)^{-} \Phi^{\top} K v.$$

Note that the optimal approximation of the value function is simply $\Phi\hat{\theta} = \Pi_K v$, which is the projection of v into the space spanned by the basis functions, but with respect to the (semi) inner product $\langle \cdot, \cdot \rangle_K$.

When K is chosen as K_{gtd} or K_{lstd}, $\hat{\theta}$ coincides with the *TD fixpoint* $\hat{\theta}_{td} = \left(\Phi^{\top} D L \Phi\right)^{-} \Phi^{\top} D r$. It also follows that the Bellman error corresponding to the optimal weights is

$$\hat{\varepsilon} = r - L\Phi\hat{\theta} = L\left(v - \Pi_K v\right),$$

[1] With a little abuse of terminology, we will not distinguish between a function f on $\mathcal{S}$ and an S-by-1 column vector $[f(1), \ldots, f(S)]^{\top}$.

[2] The inverse and transpose of an arbitrary matrix A is denoted A^{-} and $A^{\top}$, respectively.

and the minimum of the objective function is

$$\hat{J} = \min_{\theta} J = v^\top K v - (\Pi_K v)^\top K (\Pi_K v). \quad (3)$$

3. The ideal basis function from TD-error

Once the optimal weights have been found, no more improvement can be made to the value-function approximation with the given basis functions. To further improve performance, additional basis functions can be added. If $\Phi_+ = [\Phi : \phi]$, and $\theta_+ = [\theta^\top : 1]^\top$, where $[A : B]$ is the matrix with A and B juxtaposed, then the Bellman error becomes

$$\begin{aligned}\varepsilon_+ &= r - L\Phi_+\theta_+ = r - L\Phi\theta - L\phi \\ &= L(v - \Phi\theta - \phi) \\ &= L(L^-\varepsilon - \phi).\end{aligned}$$

Ideally, we should choose $\phi = L^-\varepsilon = v - \Phi\theta$, so that ε_+ is reduced to 0, and the value function is exactly represented by $v = \Phi_+\theta_+$.

The key insight is that $\phi = L^-\varepsilon$ is the solution of the Bellman equation $\phi = \varepsilon + \gamma P\phi$, namely, *$\phi$ is the value function of the original Markov chain when its reward function is equal to the Bellman error, ε.* Since ε is also the expectation of the TD-error, ϕ can be computed by solving a second reinforcement learning problem, in which the TD-error of the first is used as the reward in the second.

The choice of ϕ above depends on both Φ and the current weights θ, which change during learning. However, in the limit θ converges to $\hat{\theta}$, so we can remove this dependency and define the *Value Function of the Bellman Error*, V-BEBF, as $\hat{\phi} = L^-\hat{\varepsilon}$, or equivalently through the Bellman equation

$$\hat{\phi} = \hat{\varepsilon} + \gamma P\hat{\phi}. \quad (4)$$

We argue that $\hat{\phi}$ is the ideal basis function for two reasons. First, adding $\hat{\phi}$ allows the value function to be represented exactly, since by Eq.1

$$\begin{aligned}v = L^-r &= L^-\left(r - L\Phi\hat{\theta} + L\Phi\hat{\theta}\right) \\ &= L^-\left(\hat{\varepsilon} + L\Phi\hat{\theta}\right) \\ &= \hat{\phi} + \Phi\hat{\theta}.\end{aligned}$$

Second, as can also be seen from this last equation, adding $\hat{\phi}$ does not change any of the optimal weights for the existing basis functions. Therefore, when $\hat{\phi}$ is added, there is no need to re-adjust the weights learned previously. It is straightforward to verify that $\hat{\phi}$ (up to a non-zero multiplicative constant) is the only basis function possessing these two properties. Moreover,

$$0 = -\frac{1}{2}\nabla_\theta J\Big|_{\theta=\hat{\theta}} = \Phi^\top K L^-\hat{\varepsilon} = \Phi^\top K\hat{\phi}, \quad (5)$$

indicating that $\hat{\phi}$ is orthogonal to the first N basis functions with respect to the (semi) inner product $\langle\cdot,\cdot\rangle_K$.

3.1. Comparison with BEBFs

The idea of generating basis functions from Bellman error has been explored several times in the literature. In particular, Parr et al. (2007) carried out a theoretical study, where they pointed out that a sequence of normalized BEBFs, $\hat{\varepsilon}$ in our terminology[3], form an orthonormal basis in $\mathbb{R}^S$ with respect to $\langle\cdot,\cdot\rangle_D$, assuming that K_{lstd} is used in the objective function. As a consequence, repeatedly adding BEBFs eventually allows any value function to be represented exactly. A recent variation, namely the Bellman Average Reward Bases (BARBs), replaces the very first BEBF, r, with $P^\infty r$, where $P^\infty = \lim_{n\to\infty} P^n$. BARBs are equivalent to BEBFs if P is ergodic, yet demonstrate faster convergence otherwise, particularly when $\gamma \to 1$ (Mahadevan and Liu, 2010).

The formulation of V-BEBF bears a clear resemblance to BEBF in that it is also based on the Bellman error. However, there is a key difference. In the worst case, representing a value function requires *a whole sequence* of BEBFs, even if all BEBFs are computed exactly. In contrast, a single additional V-BEBF, if computed exactly, is sufficient to represent the value function. In fact, when the initial basis function set is empty, the first V-BEBF is simply L^-r, the value function itself.

The difference between V-BEBF and BEBF is captured by the following proposition (the proof is deferred to the appendix), and illustrated by the example in Figure.3.1.

Proposition 1 *Consider the objective function J as defined in Eq.2, with K an arbitrary fixed positive definite matrix. Let $\hat{J}$ and $\hat{J}_+$ be the minimum of J as defined in Eq.3 corresponding to the basis functions Φ and $[\Phi : \hat{\varepsilon}]$. Then*

$$\rho = \frac{\hat{J}_+}{\hat{J}} \le \gamma^2,$$

with the equality holding iff r is chosen such that

1. $\Pi_K P v = \Pi_K P\Phi\hat{\theta}$;
2. $\hat{\phi}^\top K\hat{\phi} = \hat{\phi}^\top P^\top K P\hat{\phi}$;
3. $\hat{\phi}^\top K P\hat{\phi} = \gamma\hat{\phi}^\top K\hat{\phi}$.

[3] The sequence of BEBFs is constructed iteratively, such that $\phi_1 = r$, and $\phi_{k+1} = \hat{\varepsilon}^{(k)} = r - L\Phi^{(k)}\hat{\theta}^{(k)}$, where $\Phi^{(k)} = [\phi_1, \cdots, \phi_k]$ and $\hat{\theta}^{(k)}$ are the optimal weights.

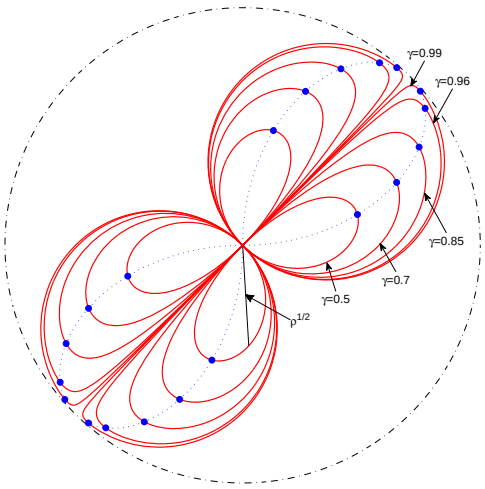

Figure 1. Consider a simple two-state MDP with transition matrix $P = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$, and let $K = L^\top DL$, so that J is the MSBE. Assume initially there are no basis functions. In this case the first BEBF is r, and the V-BEBF $L^- r$ is the value function itself. The figure shows how $\rho^{\frac{1}{2}}$ varies with γ and r. When r varies on the unit circle (dotted black circle), the distance between the points on the butterfly-shaped curve and the origin denotes $\rho^{\frac{1}{2}}$, and each curve corresponds to a different γ. The blue dots show where ρ achieves its maximum (computed from Proposition 1). It can be seen that when γ approaches 1, the worst-case ρ approaches the unit circle, which indicates less and less improvement when the BEBF is added. Note that ρ is always zero when r is exactly on $\left[\cos\frac{\pi}{4}, \sin\frac{\pi}{4}\right]^\top$ and $\left[\cos\frac{3\pi}{4}, \sin\frac{3\pi}{4}\right]^\top$, the eigendirections of L, but changes abruptly when r leaves the first eigendirection.

In particular, when K is chosen as either $L^\top DL$ or D, ρ corresponds to the *Mean Square Bellman error* (MSBE) and the *Mean Square Value Error* (MSVE), respectively, and the fraction of improvement from adding a BEBF is only $1-\gamma^2$ in the worst case. However, if the V-BEBF is added as the new basis function, the new minimum of J is always 0 since the value function is represented exactly.

An analysis in a similar spirit was proposed by Mahadevan and Liu (2010), who showed that the error in approximating the value function using the first m BEBFs is bounded in terms of the Chebyshev polynomial of degree m and the condition number of L. Our result, albeit less general, is simpler to interpret and allows construction of the worst-case scenario.

3.2. Approximating V-BEBF

In theory, the V-BEBF can be computed as the value function of the Markov chain using TD-error as the reward, provided that the weights for the current basis functions are set optimally. In practice, the V-BEBF has to be approximated and three sources of errors are anticipated. First, the exact representation of V-BEBF (as well as BEBF) requires storage of size S, which is not available in the first place. Therefore, function approximation must be used. Second, the convergence of θ to $\hat{\theta}$ happens only asymptotically, and thus only the TD-errors, ε, corresponding to the current θ can be used as the reward. Finally, error arises when estimating V-BEBF from a finite set of samples. Formally, let $\mathcal{B}$ be the set of functions from which the representations of V-BEBF approximators are chosen, and let

$$\hat{\phi}_\theta = \arg\min_{\phi\in\mathcal{B}} \|\phi - L^-\varepsilon\|$$

be the best-in-class given θ, then the error between an estimate ϕ and $\hat{\phi}$, the exact V-BEBF, is bounded by

$$\left\|\phi - \hat{\phi}\right\| \le \left\|\hat{\phi}_\theta - L^-\varepsilon\right\| + \left\|\Phi\hat{\theta} - \Phi\theta\right\| + \left\|\phi - \hat{\phi}_\theta\right\|,$$

with the three items on the right hand side corresponding to the three sources of error. As a result, though only one V-BEBF is needed in principle, in practice we still rely on repeatedly adding new approximations of V-BEBFs to compensate for the error in the estimation, as well as changes in the policy and non-stationarities in the environment.

4. Incremental basis projection with V-BEBF

The result in the previous section suggests a way to improve value function approximation incrementally, whereby a primary reinforcement learner receives reward from the environment, modifies its value function estimate over a set of basis functions, and propagates the TD-error to a secondary reinforcement learner, which estimates the value function of the TD-error, i.e., an approximation of V-BEBF, which then becomes the new basis function used by the primary learner.

The secondary learner can use any form of approximator for V-BEBF. However, in this paper, V-BEBF is combined with a simple linear architecture, which we refer to as *linear basis projection* (LBP). In LBP, the primary reinforcement learner uses a set of N 'refined' basis functions $\Phi = [\phi_1, \cdots, \phi_N]$, that are linear combinations of a set of M 'raw' basis functions, $\Psi = [\psi_1, \cdots, \psi_M]$, where $N \ll M$, such that

$$\phi_n = \sum_{m=1}^{M} w_{m,n}\psi_m = \Psi w_n,$$

where $w_n = [w_{1,n}, \ldots, w_{M,n}]^\top$ are the mixing coefficients. Let $W = [w_1, \ldots, w_N]$, then $\Phi = \Psi W$. The objective is then to build W. The main advantage of

LBP is that it reduces the prior knowledge required in designing the basis functions: one may simply generate a large set of raw basis functions, with the hope that it is representative enough for the task even though each single basis function may be only marginally related. In particular, Ghavamzadeh et al. (2010) considered the theoretical property when W are chosen i.i.d. Gaussian; Boots and Gordon (2010) proposed a method where W is chosen to retain maximum predictive information. In addition, basis function selection under sparsity constraints (Kolter and Ng, 2009) can also be seen as a special case where the W are sparse binary matrices with each w_n containing exactly one non-zero entry.

Our approach combines V-BEBF with LBP, in the sense that mixing coefficients w are generated by the secondary learner such that Ψw approximates the V-BEBF, and w is added as a new column of W, which amounts to adding V-BEBF as a new refined basis function. We refer to this new approach Incremental Basis Projection with V-BEBF, or IBP-V.

4.1. Batch IBP-V

We explore the case where the primary and the secondary learner use LSTD[4]. Let $\hat{\theta}_{raw} = (\Psi^\top D L \Psi)^- \Psi^\top D r$ be the LSTD solution using the raw basis functions, $\hat{\theta}_{ref} = (\Phi^\top D L \Phi)^- \Phi^\top D r$ be the LSTD solution using the refined basis functions. Then by definition the solution of V-BEBF is given by

$$\hat{w}_{vbebf} = \left(\Psi^\top D L \Psi\right)^- \Psi^\top D \hat{\varepsilon}_{ref} \tag{6}$$

$$= \hat{\theta}_{raw} - W\hat{\theta}_{ref}, \tag{7}$$

where $\hat{\varepsilon}_{ref} = r - L\Psi W \hat{\theta}_{ref}$ is the Bellman error. Note that $\hat{\theta}_{raw} = \hat{w}_{vbebf} + W\hat{\theta}_{ref}$, which indicates that a single LSTD solution of the V-BEBF allows the exact representation of $\hat{\theta}_{raw}$. Also, note that the LSTD solution of the BEBF can be derived by just replacing $(\Psi^\top D L \Psi)^-$ with $(\Psi^\top D \Psi)^-$ in Eq.6.

This naive implementation of IBP-V offers no computational advantage over directly computing $\hat{\theta}_{raw}$, since starting from an empty W, the first V-BEBF is exactly $\hat{\theta}_{raw}$. Without assumptions about the sparsity of the basis functions, the exact computation of $\hat{\theta}_{raw}$ requires $O\left(M^2 T\right)$ time and $O\left(M^2\right)$ storage (Geramifard et al., 2006), which is often not feasible. For this reason, we consider using only $B \ll M$ randomly chosen raw basis functions to approximate each V-BEBF (see Algorithm 1). If N V-BEBFs are constructed in total to approximate the value function, then the overall computational cost is $O(MT) + O\left(NB^2T\right) + O\left(N^3T\right)$, where the three terms correspond respectively to the cost of computing M raw basis functions, N V-BEBFs, and the weights over the refined basis functions for N times. The storage cost is $O(NB) + O\left(B^2\right) + O\left(N^2\right)$, with the first term being the storage for W.

[4]For succinctness, we use the exact sample distribution D and transition matrix P. But in practice, both D and P must be replaced by their finite sample approximations.

Algorithm 1 Batch-IBP-V[5]

Input: samples $(s_t, s_{t+1}, r_t)_{t=1}^T$, discount factor γ, raw basis functions $\psi_1, \ldots, \psi_M$, number of basis functions for each V-BEBF B, maximum number of refined basis functions N

Output: mixing matrix W, weights over refined basis functions θ

$W \leftarrow []$; $\theta \leftarrow []$
while $n < N$ **do**
  Select $U \subset \{1, \ldots, M\}$ with $|U| = B$
  for $t = 1$ to T **do**
    $\omega \leftarrow [\psi_1(s_t), \ldots, \psi_M(s_t)]$
    $\omega_p \leftarrow [\psi_1(s_{t+1}), \ldots, \psi_M(s_{t+1})]$
    $\delta_t \leftarrow r_t + \gamma\omega_p W\theta - \omega W\theta$
  end for
  $w_r \leftarrow \text{LSTD}\left((s_t, s_{t+1}, \delta_t)_{t=1}^T, \gamma, \Psi_{[U]}\right)$
  $w \leftarrow 0_{M\times 1}$; $w_{[U]} \leftarrow w_r$; $W \leftarrow [W \vdots w]$
  $\theta \leftarrow \text{LSTD}\left((s_t, s_{t+1}, r_t)_{t=1}^T, \gamma, \Psi W\right)$
  $n \leftarrow n + 1$
end while

Algorithm 1 raises the issue of how to choose N, the number of refined basis functions to construct, and B, the number of raw basis functions used to generate each refined basis functions. The following analysis shows that we can choose $N = B = (cM)^{\frac{1}{2}}$, where $c = -\log\epsilon$, with the guarantee that at least $1-\epsilon$ fraction of the raw basis functions are covered when constructing the refined basis functions. In addition, with this choice of N, B, the computational complexity becomes $O\left((cM)^{\frac{3}{2}} T\right)$ in time and $O\left(c^2 M\right)$ in storage. To see this, note that the probability of a raw basis function getting selected at least once is $1 - \left(1 - \frac{B}{M}\right)^N$, and our assumption requires

$$1 - \left(1 - \frac{B}{M}\right)^N \geq 1 - \epsilon, \text{ or } N \geq \frac{-\log\epsilon}{-\log\left(1 - \frac{B}{M}\right)}.$$

Assume $\frac{B}{M} = o(1)$, and note that $-\log(1-x) > x$. It suffices that $NB > cM$, and letting $N = B = (cM)^{\frac{1}{2}}$ gives the complexity result.

[5]We assume the subroutine LSTD($samples, \gamma, basis$) produces LSTD estimates of the weights for the provided samples, i.e., triples (s_t, s_{t+1}, r_t), and basis functions. Also, for an arbitrary set of integers U, $w_{[U]}$ denotes the subvector with entries indexed by U.

Algorithm 2 Online-IBP-V[6]

Input: sample trajectory (s_t, s_{t+1}, r_t), discount factor γ, raw basis functions $\psi_1, \ldots, \psi_M$, steps before adding each V-BEBF C
Output: mixing matrix W, weights over refined basis functions θ

1: $c \leftarrow C$
2: $W \leftarrow []$; $\theta \leftarrow []$; $w \leftarrow 0_{M\times 1}$
3: **repeat**
4: $\quad \omega \leftarrow [\psi_1(s_t), \ldots, \psi_M(s_t)]$
5: $\quad \omega_p \leftarrow [\psi_1(s_{t+1}), \ldots, \psi_M(s_{t+1})]$
6: $\quad \delta \leftarrow r_t + (\gamma\omega_p - \omega) W\theta$
7: $\quad \theta \leftarrow \theta + \alpha\delta(\omega W)^\top$
8: $\quad w \leftarrow w + \alpha(\delta + \gamma\omega_p w - \omega w)\omega^\top$
9: $\quad c \leftarrow c - 1$
10: $\quad$ **if** $c = 0$ **then**
11: $\quad\quad c \leftarrow C$
12: $\quad\quad W \leftarrow [W : w]$; $\theta \leftarrow [\theta^\top : 0]^\top$; $w \leftarrow 0_{M\times 1}$
13: $\quad$ **end if**
14: **until** trajectory ends

4.2. Online IBP-V

Batch IBP-V can be modified to work online, e.g., by replacing LSTD with iLSTD (Geramifard et al., 2006). One may instead use the linear complexity methods such as TD or TDC (Sutton et al., 2009; Maei and Sutton, 2010), (see Algorithm 2), so that the complexity of each time step is $O(MN)$ if all raw basis functions are used to estimate the V-BEBF or BEBF, or $O(BN)$ if only B raw basis functions are used. In practice, N can be controlled by dynamically removing refined basis functions whose weights are very small. If N is upper bounded by a constant, then online IBP-V is linear in the number of raw basis functions. It is also worth mentioning that this structure offers additional advantages in terms of convergence speed due to the low dimensionality of the resulting primary reinforcement learning problem.

5. Experiments

Batch and online IBP-V were tested on randomly generated[7] Markov chains with 500 states and a branching factor (the number of successor states) of 5. All experiments used binary raw basis functions over the states, that were generated by filling Ψ with i.i.d. Bernoulli variables with $p = 0.2$. The error was measured using

$$\eta(W,\theta) = \frac{\left(\hat{\theta}_{raw} - W\theta\right)^\top \Psi^\top D \Psi \left(\hat{\theta}_{raw} - W\theta\right)}{\hat{\theta}_{raw}^\top \Psi^\top D \Psi \hat{\theta}_{raw}}, \quad (8)$$

which is the normalized MSVE with respect to the best possible value function approximation $\Psi\hat{\theta}_{raw}$ from the current set of raw basis functions. The initial W is set to empty so that the initial value of η is always 1.

For the batch case, IBP-V was compared with its BEBF counterpart, termed IBP-B, and random feature projection, described by Ghavamzadeh et al. (2010). The number of raw basis functions, B, used to compute each V-BEBF or BEBF, and the number of refined basis functions N generated in total, were both set to $\sqrt{5M}$.

For the online case, IBP-V was compared to IBP-B with both $B = M$ (each refined basis function uses all of the raw basis functions), and $B = M/2$ (each refined basis function uses half of the randomly selected raw basis functions). A refined basis function was added after every 5000 time steps. To provide a baseline, these four incremental approaches were compared to TD using only the raw basis functions. All TD updates used the same learning rate 0.05.

For all experiments, each method was run 100 times with $\gamma = \{0.9, 0.99, 0.99\}$ and $M = \{200, 1000\}$ raw basis functions, for a total of six different comparisons in both the batch and online settings.

5.1. Results: Batch

Figure 2 plots the error, η, against the number of refined basis functions added. Each curve is the average of 100 runs, each of which is based on an independent sample of the Markov chain and the set of raw basis functions, and a trajectory of length $T = 5000$ is used. (The error bar is small thus is omitted). It can be seen that both IBP-V and IBP-B perform significantly better than random feature projection. It is also clear that when γ approaches 1, the difference between adding IBP-V and IBP-B increases dramatically—when $\gamma = 0.999$, the error decreases extremely slowly as more BEBFs are added, which coincides with Proposition 1. For smaller γ, the difference between BEBF and V-BEBF diminishes, since BEBF can be seen as V-BEBF with discount factor 0. In addition, although each V-BEBF is constructed from a much smaller number of raw basis functions, (e.g., $B = 70$ for $M = 1000$), the error still decreases rapidly as the number of V-BEBFs is increased, and our heuristic choice of B and N is shown to be quite effective.

[6] For simplicity, assume TD is used in both the primary and secondary learner (see line 7 and 8), which can be replaced by other algorithms. Also, the learning rates are fixed to α.

[7] We use the method described in http://webdocs.cs.ualberta.ca/~sutton/RandomMDPs.html

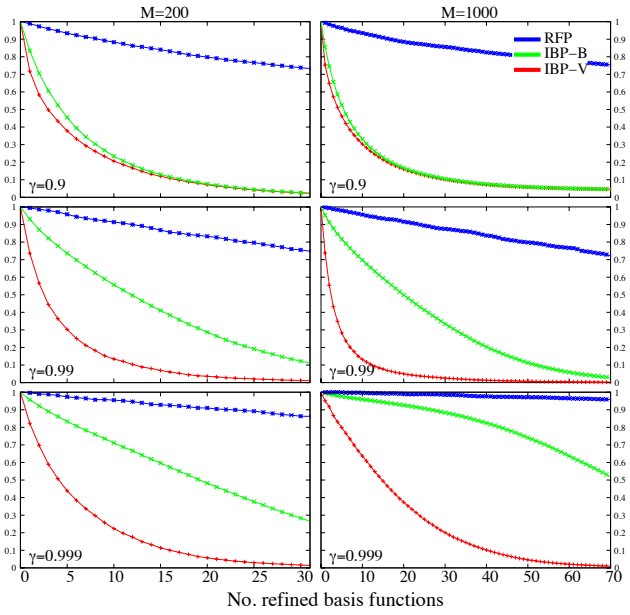

Figure 2. **Results for batch IBP**. Each graph shows the performance of the three methods compared, IBP-V, IBP-B, and random feature projection (RFP), in terms of the error η (Eq.8) against the number of refined basis function added (averaged over 100 runs).

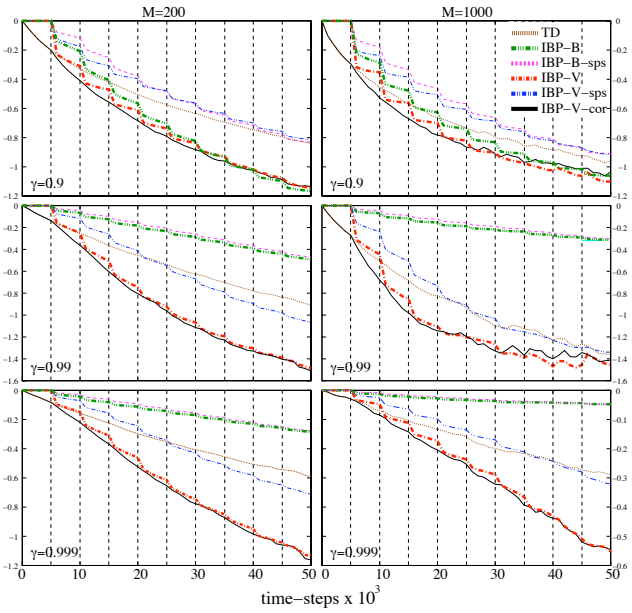

Figure 3. **Results for online IBP**. Each graph shows the performance of the six methods compared, IBP-V/B ($B = M$), IBP-V/B ($B = M/2$, tailed '-sps'), TD, and IBP-V with correction using Eq.6 (tailed '-cor'), in terms of $\log_{10} \eta$ against time (averaged over 100 runs). The vertical dotted lines indicate the moments at which a new refined basis function is added. Notice that each time a new basis function is added, the error drops abruptly.

5.2. Results: Online

Figure 3, plots the log error, $\log_{10} \eta$, against time-steps. Note the 'stair' shape of the curves for both IBP-V and IBP-B caused by the abrupt error drop each time a new refined basis function is added. These curves can be smoothened by combining the weights for both the refined basis functions and the current V-BEBF, using Eq.6. It can be seen that when the discount factor approaches 1, the first few V-BEBFs added are far more effective than the BEBFs added. Also, it is curious that IBP-V outperforms the baseline TD method after only a few basis functions are added. A possible explanation is the combined effect of the low dimensionality of the primary learner, and the less correlated weights over the different V-BEBFs, by Eq.5. In addition, the version with $B = M/2$ yields similar performance as compared to the TD-method which uses all of the raw basis functions.

6. Conclusion

We have presented a novel and powerful approach for basis-function generation, and have demonstrated its potential in combination with incremental basis projection. The result opens a new direction in automatic basis construction and can be extended in various ways: The general idea of using V-BEBF, the 'Value Function of the TD-error', as basis functions is not restricted to linear architectures nor to Markovian processes. Both batch and online IBP-V show promising performance, even with the most naive implementation, which suggests substantial room for further development.

References

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *ICML'95*, 1995.

B. Boots and G. J. Gordon. Predictive state temporal difference learning. In *NIPS'10*, 2010.

J. A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:233–246, 2002. ISSN 0885-6125.

S. J. Bradtke, A. G. Barto, and L. P. Kaelbling. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

A. Geramifard, M. Bowling, and R. S. Sutton. Incremental least-squares temporal difference learning. In *AAAI'06*, 2006.

M. Ghavamzadeh, A. Lazaric, O. Maillard, and R. Munos. Lstd with random projections. In *NIPS'10*, 2010.

P. W. Keller, S. Mannor, and D. Precup. Auto-

matic basis function construction for approximate dynamic programming and reinforcement learning. In *ICML'06*, 2006.

Z. J. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *ICML'09*, 2009.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4: 1107–1149, 12 2003. ISSN 1532-4435.

H. R. Maei and R. S. Sutton. Gq(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *AGI'10*, 2010.

S. Mahadevan and B. Liu. Basis construction from power series expansions of value functions. In *NIPS'10*, 2010.

S. Mahadevan, M. Maggioni, and C. Guestrin. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8: 2007, 2006.

R. Parr, C. Painter-Wakefield, L.-H. Li, and M. Littman. Analyzing feature generation for value-function approximation. In *ICML'07*, 2007.

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

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. mit press, 1998.

R. S. Sutton, C. Szepesvári, and H. R. Maei. A convergent o(n) algorithm for off-policy temporal-difference learning with linear function approximation. In *NIPS'08*, 2008.

R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML'09*, 2009.

J.-H. Wu and R. Givan. Feature-discovering approximate value iteration methods. In *SARA'05*, pages 321–331, 2005.

Appendix

Proof of Proposition 1. Let ϕ be the new basis function, then the minimum of the objective functions $\hat{J}$ and $\hat{J}_+$ without and with ϕ added are given respectively by Eq.3 and

$$\hat{J}_+ = v^\top K v - v^\top K \left[\Phi, \phi\right] \left(\left[\Phi, \phi\right]^\top K \left[\Phi, \phi\right]\right)^- \left[\Phi, \phi\right]^\top K v.$$

Change variables for simplicity: Write $K = A^\top A$, $u = Av$, $\Psi = A\Phi$, $\psi = A\phi$, then $\hat{J} = u^\top (I - \Pi) u = u^\top V u$, where $\Pi = \Psi \left(\Psi^\top \Psi\right)^- \Psi^\top$, and $V = I - \Pi$. Note that $\Pi = \Pi^2$, $V = V^2$. From block matrix inversion and Kailath variant,

$$\begin{bmatrix} \Psi^\top \Psi & \Psi^\top \psi \\ \psi^\top \Psi & \psi^\top \psi \end{bmatrix}^- = \begin{bmatrix} \left(\Psi^\top \Psi\right)^- & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\psi^\top V \psi} v v^\top,$$

where $v = \left[-\psi^\top \Psi \left(\Psi^\top \Psi\right)^-, 1\right]^\top$, therefore

$$\hat{J}_+ = u^\top V u - \frac{u^\top V \psi \cdot \psi^\top V u}{\psi^\top V \psi}.$$

Note that here the additional basis function is the BEBF, so $\phi = \hat{\varepsilon} = LA^- V u$, therefore

$$\begin{aligned} \rho = \frac{\hat{J}_+}{\hat{J}} &= 1 - \frac{\left(u^\top V \psi\right)\left(\psi^\top V u\right)}{\left(\psi^\top V \psi\right)\left(u^\top V u\right)} \\ &= 1 - \frac{\left(u^\top V A L A^- V u\right)^2}{\left(u^\top V A^{-\top} L^\top A^\top V A L A^- V u\right)\left(u^\top V u\right)} \\ &= 1 - \frac{\left(z^\top K L z\right)^2}{\left(z^\top L^\top A^\top V A L z\right)\left(z^\top K z\right)}, \end{aligned}$$

with $z = A^- V u = A^- V A L^- r$. Also,

$$A^\top V A = A^\top \left(I - \Psi \left(\Psi^\top \Psi\right)^- \Psi^\top\right) A \prec A^\top A = K,$$

thus

$$\rho \le 1 - \frac{\left(z^\top K L z\right)^2}{\left(z^\top L^\top K L z\right)\left(z^\top K z\right)}.$$

And the equality holds if and only if

$$\left(I - V\right) A L z = \left(I - V\right) A L A^- \cdot V A L^- r = 0.$$

Simplify this equation gives the first condition.

Now assume that the equality holds, and expand $L = I - \gamma P$, then

$$\begin{aligned} \rho &= 1 - \frac{\left(z^\top K z - \gamma z^\top K P z\right)^2}{z^\top K z \cdot z^\top \left(I - \gamma P\right)^\top K \left(I - \gamma P\right) z} \\ &= \gamma^2 \cdot \frac{\left(z^\top K z\right)\left(z^\top P^\top K P z\right) - \left(z^\top K P z\right)^2}{z^\top K z \cdot z^\top \left(I - \gamma P\right)^\top K \left(I - \gamma P\right) z}. \end{aligned}$$

Write $x = Az$, $y = APz = APA^- x$, then

$$\rho = 1 - \frac{\langle x, x - \gamma y\rangle^2}{\|x\|^2 \|x - \gamma y\|^2} = \gamma^2 \frac{\langle x, x\rangle \langle y, y\rangle - \langle x, y\rangle^2}{\|x\|^2 \|x - \gamma y\|^2},$$

where $\langle x, y\rangle = x^\top y$ and $\|x\|^2 = \langle x, x\rangle$. We show that

$$\|x\|^2 \|x - \gamma y\|^2 \ge \|x\|^2 \|y\|^2 - \langle x, y\rangle^2.$$

Indeed,

$$\begin{aligned} &\|x\|^2 \|x - \gamma y\|^2 - \|x\|^2 \|y\|^2 + \langle x, y\rangle^2 \\ &= \left(\gamma \|x\|^2 - \langle x, y\rangle\right)^2 + \left(1 - \gamma^2\right) \|x\|^2 \left(\|x\|^2 - \|y\|^2\right), \end{aligned}$$

and notice that $y = APA^- x$ and P is a transition matrix, so all the eigenvalues of APA^- are smaller or equal than 1, therefore $\|y\|^2 \le \|x\|^2$. Also, the equality holds iff: a) $\|x\|^2 = \|y\|^2$ and b) $\langle x, y\rangle = \gamma \|x\|^2$, which correspond to the second and the third condition. ■