

Adaptive Bases for Reinforcement Learning

Dotan Di Castro and Shie Mannor

Department of Electrical Engineering
Technion - Israel Institute of Technology
`{dot,shie}@{tx,ee}.technion.ac.il`

Abstract. We consider the problem of reinforcement learning using function approximation, where the approximating basis can change dynamically while interacting with the environment. A motivation for such an approach is maximizing the value function fitness to the problem faced. Three errors are considered: approximation square error, Bellman residual, and projected Bellman residual. Algorithms under the actor-critic framework are presented, and shown to converge. The advantage of such an adaptive basis is demonstrated in simulations.

1 Introduction

Reinforcement Learning (RL) [4] is an approach for solving Markov Decision Processes (MDPs), when interacting with an unknown environment. One of the main obstacles in applying RL methods is how to cope with a large state space. In general, the underlying methods are based on dynamic programming, and include adaptive schemes that mimic either value iteration, such as Q-learning, or policy iteration, such as Actor-Critic (AC) methods. While the former attempt to directly learn the optimal value function, the latter are based on quickly learning the value of the currently used policy, followed by a slower policy improvement step. In this paper we focus on AC methods.

There are two major problems when solving MDPs with a large state space. The first is the storage problem, i.e., it is impractical to store the value function and the optimal action explicitly for each state. The second is generalization: some notion of similarity between states is needed since most states are not visited or visited only a few times. Thus, these issues are addressed by the Function Approximation (FA) approach [4], that involves approximating the value function by functional approximators with a smaller number of parameters in comparison to the original number of states. The success of this approach rests mainly on selecting appropriate features, and on a proper choice of the approximation architecture. In a linear approximation architecture, the value of a state is determined by linear combination of the low dimensional feature vector. In the RL context, linear architectures enjoy convergence results and performance guarantees (e.g., [4]).

The approximation quality depends on the choice of the basis functions. In this paper we consider the possibility of tuning the basis functions on-line, under the AC framework. As mentioned before, an agent interacting with the environment is composed of two sub-systems. The first is a critic, that estimates

the value function for the states encountered. This sub-system acts on a fast time scale. The second is an actor, that based on the critic output, and mainly the *temporal-difference* (TD) signal, improves the agent’s policy using gradient methods. The actor operates on a second time scale, slower than the time-scale of the critic. Bhatnagar et al. [5] proved that such an algorithm with an appropriate relation between the time scales, converges.

We suggest to add a third time scale that is slower than both the critic and the actor, minimizing some error criteria while adapting the critic’s basis functions to better fit the problem. Convergence of the value function, policy and the basis is guaranteed in such an architecture, and simulations show that a dramatic improvement can be achieved using basis adaptation.

Using multiple time scales may pose a convergence drawback at first sight. Two approaches may be applied in order to overcome this problem. First, a recent work of Mokkadem and Pelletier [12], based on previous research by Polyak [13] and others, have demonstrated that combining the algorithm iterates with the averaging method of [13] leads to convergence rate in distribution that is the same as the optimal rate. Second, in multiple time scales the rate between the time steps of the slower and faster time scales should converge to 0. Thus, time scales which are close, operate on the fast time scale, and satisfy the condition above, are easy to find for any practical needs.

There are several works done in the area of adaptive bases. These works do not address the problem of policy improvement with adaptive bases. We mention here two noticeable works which are similar in spirit to our work. The first work is of Menache et al. [11]. Two algorithms were suggested for adaptive bases by the authors: one algorithm is based on gradient methods for *least-squares TD* (LSTD) of Bardtke and Barto [2], and the other algorithm is based on the cross entropy method. Both algorithms were demonstrated in simulations to achieve better performance than their fixed basis counterparts but no convergence guarantees were supplied. Yu and Bertsekas [19] suggested several algorithms for two main problem classes: policy evaluation and optimal stopping. The former is closer to our work than the latter so we focus on this class. Three target functions were considered in that work: mean TD error, Bellman error, and projected Bellman error. The main difference between [19] and our work (besides the policy improvement) is the following. The algorithmic variants suggested in [19] are in the flavor of LSTD and LSPE algorithms [3], while in our work the algorithms are TD based, thus, in our work no matrix inversion is involved. Also, we demonstrate the effectiveness of the algorithms in the current work.

The paper is organized as follows. In Section 2 we define some preliminaries and outline the framework. In Section 3 we introduce the algorithms suggested for adaptive bases. In Section 4 we show the convergence of the algorithms suggested, while in Section 5 we demonstrate the algorithms in simulations. In Section 6 we discuss the results.

2 Preliminaries

In this section, we introduce the framework, review actor-critic algorithms, overview multiple time scales stochastic approximation (MTS-SA), and state a related theorem which will be used later in proving the main results.

2.1 The Framework

We consider an agent interacting with an unknown environment that is modeled by a *Markov Decision Process* (MDP) [14] in discrete time with a finite state set X and an action set U where $N \triangleq |X|$. Each selected action $u \in U$ of the agent determines a stochastic transition matrix $P_u = [P_u(y|x)]_{x,y \in X}$, where y is the state followed the state x .

For each state $x \in X$ the agent receives a corresponding reward $g(x)$ that depend only on the current state¹. The agent maintains a parameterized *policy function* which is a probabilistic function, denoted by $\mu_\theta(u|x)$, mapping an observation $x \in X$ into a probability distribution over the controls U . The parameter $\theta \in \mathbb{R}^{K_\theta}$ is a tunable parameter where $\mu_\theta(u|x)$ is a differentiable function w.r.t. θ . We note that for different θ 's, different probability distributions over \mathcal{U} may be associated for each $x \in \mathcal{X}$. We denote by $x_0, u_0, g_0, x_1, u_1, g_1, \dots$ a state-action-reward trajectory where the subindex specifies time.

Under each policy induced by $\mu_\theta(u|x)$, the environment and the agent induce together a Markovian transition function, denoted by $P_\theta(y|x)$, satisfying $P_\theta(y|x) = \sum_u \mu_\theta(u|x) P_u(y|x)$. The Markovian transition function $P_\theta(y|x)$ induces a stationary distribution over the state space X , denoted by $D(\theta)$. This distribution induces a natural norm, denoted by $\|\cdot\|_{D(\theta)}$, which is a weighted norm and is defined by $\|x\|_{D(\theta)}^2 \triangleq x^\top D(\theta) x$. Note that when the parameter θ changes, the norm changes as well. We denote by $E_\theta[\cdot]$ the expectation operator w.r.t. the measures $P_\theta(y|x)$ and $D(\theta)$. There are several performance criteria investigated in the RL literature that differ mainly on their time horizon and the treatment of future rewards [4]. In this work we focus on *average reward* criteria defined by

$$\eta_\theta = E_\theta[g(x)]. \quad (1)$$

The agent's goal is to find the parameter θ that maximizes η_θ . Similarly, define the (*differential*) *value function* as

$$J(x) \triangleq E_\theta \left[\sum_{n=0}^{\tau} (g(x_n) - \eta_\theta) \middle| x_0 = x \right], \quad (2)$$

where $\tau \triangleq \min\{k > 0 | x_k = x^*\}$ and x^* is some recurrent state for all policies, we assume to exist. Define the *Bellman operator* as $TJ(x) = r - \eta + E_\theta[J(y)|x]$. Thus, based on (2) it is easy to show the following connection between the average reward to the value function under a given policy [3], i.e.,

$$J(x) = g(x) - \eta + E_\theta[J(y)|x] \triangleq TJ(x), \quad (3)$$

¹ Generalizing the results presented here to state-action rewards is straight forward.

For later use, we denote by TJ and J the column representations of $J(x)$ and $TJ(x)$ respectively.

We define the Temporal Difference (TD) [4,16] of the state x followed by the state y as $d(x, y) = g(x) - \eta + J(y) - J(x)$, where for a specific time n we abbreviate $d(x_n, x_{n+1})$ as d_n . Based on (3) we can see that

$$\mathbb{E}_\theta[d(x, y)|x] = 0, \quad \text{and} \quad \mathbb{E}_\theta[d(x, y)] = 0. \quad (4)$$

Based on this property, a wide family of algorithms known as TD algorithm exist [4], where common to all these algorithms is solving (4) iteratively.

Notational comment: from now on, we omit the dependency on θ whenever it is clear from the context.

2.2 Actor-Critic Algorithms

A well known class of RL approaches is the so called actor-critic (AC) algorithms, where the agent is divided into two components, an actor and a critic. The critic functions as a state value estimator using the so called *TD-learning* algorithm, whereas the actor attempts to select actions based on the TD signal estimated by the critic. These two components solve their own optimization problems separately interacting with each other.

The critic typically uses a function approximator which approximates the value function in a subspace of a reduced dimension \mathbb{R}^{K_r} . Define the basis matrix

$$\Phi \triangleq [\phi_k(x_n)]_{1 \leq n \leq N, 1 \leq k \leq K_r} \in \mathbb{R}^{N \times K_r}, \quad (5)$$

where its columns span the subspace \mathbb{R}^{K_r} . Thus, the approximation to the value function is $\tilde{J}(x, r) \triangleq \phi(x)^\top r$, where r is the solution of the following quadratic program $r = \arg \min_{r' \in \mathbb{R}^{K_r}} \|\Phi r' - J\|_D^2$. This solution yields the linear projection operator,

$$\Pi = \Phi (\Phi^\top D_\theta \Phi)^{-1} \Phi^\top D_\theta \quad (6)$$

that satisfies

$$\tilde{J}(r) = \Pi J. \quad (7)$$

where $\tilde{J}(r)$ is the vector representation of $\tilde{J}(x, r)$. Abusing notation, we define the (state dependent) projection operator on $J(x)$ as $\tilde{J}(x) = \Pi J(x)$.

As mentioned above, the actor receives the TD signal from the critic, where based on this signal, the actor tries to select the optimal action. As described in Section 2.1, the actor maintains a policy function $\mu_\theta(u|x)$. In the following, we state a theorem that serves as the foundation for the policy gradient algorithm described later. The theorem relates the gradient w.r.t. θ of the average reward, $\nabla_\theta \eta_\theta$, to the TD signal, $d(x, y)$. Define the *likelihood ratio derivative* as $\psi_\theta(x, u) \triangleq \nabla_\theta \mu_\theta(u|x) / \mu_\theta(u|x)$. We omit the dependency of ψ on x , u , and θ through that paper. The following assumption states that ψ is bounded.

Assumption 1. For all $x \in X$, $u \in U$, and $\theta \in \mathbb{R}^{K_\theta}$, there exists a positive constant, B_ψ , such that $\|\psi\|_2, \|\nabla_\theta \psi\|_2 \leq B_\psi < \infty$.

Based on this, we present the following lemma that relates the gradient of η to the TD signal [5].

Lemma 2. *The gradient of the average reward (w.r.t. to θ) can be expressed by $\nabla_{\theta}\eta = \mathbb{E}[\psi_{\theta}(x, u)d(x, y)]$.*

2.3 Multiple Time Scales Stochastic Approximation

Stochastic approximation (SA), and in particular the ODE approach [9], is a widely used method for investigating the asymptotic behavior of stochastic iterates. For example, consider the following stochastic iterate

$$\varphi_{n+1} = \varphi_n + \alpha_n G(\varphi_n, \zeta_{n+1})$$

where $\{\zeta_{n+1}\}$ is some random process and $\{\alpha_n\}$ are step sizes that form a positive series satisfying conditions to be defined later. The key idea of the technique is the following. Suppose that the iterate can be decomposed into a mean function, denoted by $F(\cdot)$, and a noise term (martingale difference noise), denoted by M_{n+1} ,

$$\varphi_{n+1} = \varphi_n + \alpha_n G(\varphi_n, \zeta_{n+1}) = \varphi_n + \alpha_n (F(\varphi_n) + M_{n+1}), \quad (8)$$

and suppose that the effect of the noise weakens due to repeated averaging. Consider the following ODE which is a continuous version of φ and $F(\cdot)$

$$\dot{\varphi}_t = (F(\varphi_t)), \quad (9)$$

where the dot above a variable stands for a time derivative. Then, a typical result of the ODE method in the SA theory suggests that the asymptotic limit of (8) and (9) are identical.

The classical theory of SA considers an iterate, which may be in some finite dimensional Euclidean space. Sometimes, we need to deal with several multidimensional iterates, dependent one on the other, and where each iterate operates on different timescale. Surprisingly, this type of SA, called *multiple time scale SA* (MTS-SA), is sometimes easier to analyze, with respect to the same iterates operate on single timescale. The first analysis of two time-scales SA algorithms was given by Borkar in [6] and later expanded to MTS by Leslie and Collins in [10]. In the following we describe the problem of MTS-SA, state the related ODEs, and finally state the conditions under which MTS-SA iterates converge. We follow the definitions of [10].

Consider L dependent SA iterates as the following

$$\varphi_{n+1}^{(i)} = \varphi_n^{(i)} + \alpha_n^{(i)} \left(F^{(i)} \left(\varphi_n^{(1)}, \dots, \varphi_n^{(N)} \right) + M_{n+1}^{(i)} \right), \quad 1 \leq i \leq L, \quad (10)$$

where $\varphi_n^{(i)} \in \mathbb{R}^{d_i}$, and $F^{(i)} : \mathbb{R}^{\otimes_{j=1}^L d_j} \rightarrow \mathbb{R}^{d_i}$. The following assumption contains a standard requirement for MTS-SA step size.

Assumption 3. (*MTS-SA step size assumptions*)

1. For $1 \leq n \leq L$, we have $\sum_{n=0}^{\infty} \alpha_n^{(i)} = \infty$, $\sum_{n=0}^{\infty} \left(\alpha_n^{(i)}\right)^2 < \infty$,
2. For $1 \leq n \leq L-1$, we have $\lim_{n \rightarrow \infty} a_n^{(i)} / a_n^{(i+1)} = 0$.

We interpret the second requirement in the following way: the higher the index i of an iterate, it operates on higher time scale. This is because that there exists some n_0 such that for all $n > n_0$ the step size of the i -th iterate is larger uniformly than the step size of the iterates $1 \leq j \leq i-1$. Thus, the i -th iterate advances more than any of the iterates $1 \leq j \leq i-1$, or in other words, it operates on faster time scale. The following assumption aggregates the main requirement for the MTS-SA iterates.

Assumption 4. (*MTS-SA iterate assumptions*)

1. $F^{(i)}(\cdot)$ are globally Lipschitz continuous,
2. For $1 \leq i \leq L$, we have $\sup_n \|\varphi_n^{(i)}\| < \infty$.
3. For $1 \leq i \leq L$, $\sum_{k=0}^n a_k^{(i)} M_{k+1}^{(i)}$ converges a.s.
4. (*The ODEs requirements*) Remark: this requirement is defined recursively where requirement (a) below is the initial requirement related to the L -th ODE, and requirement (b) below describes the i -th ODE system that is recursively based on the $(i+1)$ -th ODE system, going from $i = L-1$ to $i = 1$. Denote $\varphi^{(i \rightarrow j)} \triangleq (\varphi^{(i)}, \dots, \varphi^{(j)})$.
 (a) Define the L -th ODE system to be

$$\begin{cases} \dot{\varphi}_t^{(1 \rightarrow L-1)} = 0, \\ \dot{\varphi}_t^{(L)} = F^{(L)}(\varphi_t^{(1)}, \dots, \varphi_t^{(L)}), \end{cases} \quad (11)$$

and suppose the initial condition $\varphi_t^{(1 \rightarrow L-1)} \Big|_{t=0} = \varphi_0$. Then, there exists a Lipschitz continuous function $\xi^{(L)}(\varphi_0)$ such that the ODE system (11) converges to the point $(\varphi_0, \xi^{(L)}(\varphi_0))$.

- (b) Define the i -th ODE system, $i = L-1, \dots, 1$, to be

$$\begin{cases} \dot{\varphi}_t^{(1 \rightarrow i-1)} = 0, \\ \dot{\varphi}_t^{(i)} = F^{(i)}(\varphi^{(1)}, \dots, \varphi^{(i-1)}, \varphi^{(i)}, \xi^{(i+1)}(\varphi_0, \varphi^{(i)})), \end{cases} \quad (12)$$

where $\xi^{(i+1)}(\cdot, \cdot)$ is determined by the $(i+1)$ -th ODE system, and suppose the initial condition $\varphi_t^{(1 \rightarrow i-1)} \Big|_{t=0} = \varphi_0$. Then, there exists a Lipschitz continuous function $\xi^{(i)}(\varphi_0)$ such that the ODE system (12) converges to the point $(\varphi_0, \xi^{(i)})$.

The first two requirements are common conditions for SA iterates to converge. The third requirement ensures the noise term asymptotically vanishes. The fourth requirement ensures (using a recursive definition) that for each time scale i , where the slower time scales $1, \dots, i-1$ are static and where for the faster time scales $i+1, \dots, L$ there exists a function $\xi^{(j+1 \rightarrow L)}(\cdot)$ (which is the solution

of the $i + 1$ ODE system), there exists a Lipschitz convergent function. Based on these requirements, we cite the following theorem due to Leslie and Collins [10].

Theorem 5. *Consider the iterate (10) and suppose Assumption 3 and 4 hold. Then, the asymptotic behavior of the iterates (10) converge to the invariant set of the dynamic system*

$$\dot{\varphi}_t^{(1)} = F^{(1)}\left(\varphi_t^{(1)}, \xi^{(2)}\left(\varphi_t^{(1)}\right)\right), \quad (13)$$

where $\xi^{(2)}(\cdot)$ is determined by requirement 4 of Assumption 4.

3 Main Results

In this section we present the main theoretical results of the work. We start by introducing adaptive bases and show the algorithms that are derived from choosing different approximating schemes.

3.1 Adaptive Bases

The motivation for adaptive bases is the following. Consider an agent that chooses a basis for the critic in order to approximate the value function. The basis which one chooses with no prior knowledge might not be suitable for the problem at hand. A poor subspace where the actual value function is poorly supported may be chosen. Thus, one might prefer to choose a parameterized basis that has additional flexibility by changing a small set of parameters.

We propose to consider a basis that is linear in some of the parameters but has several other parameters that allow greater flexibility. In other words, we consider bases that are linear with respect to some of the terms (related to the fast time scale), and nonlinear with respect to the rest (related to the slow time scale). The idea is that most probably one does not lose from such an approach in general if it fails, but in many cases it is possible to obtain better fitness and thus a better performance, due to this additional flexibility. Mathematically,

$$\tilde{J}(x, r, s) = \phi(x, s)^\top r, \quad s \in \mathbb{R}^{K_s}, \quad (14)$$

where r is a linear parameter related to the fast time scale, and s is the non-linear parameter related to the slow time scale. In the view of (5), we note that from now on the matrix Φ depends on s , i.e., $\Phi \equiv \Phi_s$, and in matrix form we have $\tilde{J} = \Phi_s r$, but for ease of exposition we drop the dependency on s . The following assumption is needed for proving later results.

Assumption 6. *The columns of the the matrix Φ are linearly independent, $K_r < N$, and $\Phi r \neq e$, where e is a vector of 1's. Moreover, the functions $\phi(x, s)$ and $\partial\phi(x, s)/\partial s_i$ for $1 \leq i \leq K_s$ are Lipschitz in s with a coefficient L_ϕ , and bounded with coefficient B_ϕ .*

Notation comment: for ease of exposition, we drop the dependency on x_n , e.g., $\phi_n \equiv \phi(x_n, s_n)$, $g_n \equiv g(x_n)$. Denote $\phi \triangleq \phi(x, s)$, $\phi' \triangleq \phi(y, s)$ (where as in Section 2.1, y is the state followed the state x), $\phi'_n \triangleq \phi(x_{n+1}, s_n)$, $d_n \triangleq d(x_n, x_{n+1})$, and $d \triangleq d(x, y)$. Thus, $d = g - \eta + \phi'^\top r - \phi^\top r$ and $d_n = g_n - \eta_n + \phi_n'^\top r_n - \phi_n^\top r_n$.

3.2 Minimum Square Error and TD

Assume a basis parameterized as in (14). The *minimum square error* (MSE) is defined as

$$\text{MSE} = \frac{1}{2} \mathbb{E} \left[\left(\tilde{J}(x) - J(x) \right)^2 \right].$$

The gradient with respect to r is

$$\nabla_r \text{MSE} = \frac{1}{2} \mathbb{E} \left[\left(\tilde{J}(x) - J(x) \right) \phi \right] \approx \mathbb{E} [d\phi], \quad (15)$$

where in the approximation we use the bootstrapping method (see [16] for a disussion) in order to get the well known TD algorithm (i.e., substituting $J \approx T\tilde{J}$). On top of the above TD algorithm, we take a derivative with respect to s_i , $i = 1, \dots, K_s$, yielding

$$\frac{\partial \text{MSE}}{\partial s_i} = \mathbb{E} \left[\left(\tilde{J}(x) - J(x) \right) \frac{\partial \tilde{J}(x)}{\partial s_i} \right] \approx \mathbb{E} \left[d \frac{\partial \phi^\top}{\partial s_i} r \right], \quad (16)$$

where again we use the bootstrapping method. Note that this equation gives the non-linear TD procedure for the basis parameters. We use SA in order to solve the stochastic equations (15) and (16), which together with Theorem 2 is the basis for the following algorithm. For technical reasons, we add an requirement that the iterates for θ and s are bounded, which practically is not constraining (see [9] for discussion on constrained SA).

Algorithm 7. *Adaptive basis TD (ABTD).*

$$\eta_{n+1} = \eta_n + \alpha_n^{(3)} (g_n - \eta_n), \quad (17)$$

$$r_{n+1} = r_n + \alpha_n^{(3)} d_n \phi_n, \quad (18)$$

$$\theta_{n+1} = H_P^{(\theta)} \left[\theta_n + \alpha_n^{(2)} \psi_n d_n \right], \quad (19)$$

$$s_{i,n+1} = H_P^{(s)} \left[s_{i,n} + \alpha_n^{(1)} d_n \frac{\partial \phi_n^\top}{\partial s_i} r_n \right], \quad i = 1, \dots, K_s, \quad (20)$$

where $H_P^{(\theta)}$ and $H_P^{(s)}$ are projection operators into a non-empty open constraints set whenever $\theta_n \notin H_\theta$ and $s \notin H_s$, respectively, and the step size series $\{\alpha_n^{(i)}\}$ for $i = 1, 2, 3$ satisfy Assumption 3.

We note that this algorithm is an AC algorithm with three time scales: the usual two time scales, i.e., choosing $\{\alpha_n^{(1)}\}_{n=1}^\infty \equiv 0$ yields Algorithm 1 of [5], and the third iterates is added for the basis adaptation, which is the slowest.

3.3 Minimum Square Bellman Error

The Minimum Square Bellman Error (MSBE) is defined as

$$\text{MSBE} = \frac{1}{2} \mathbb{E} \left[\left(T\tilde{J}(x) - \tilde{J}(x) \right)^2 \right].$$

The gradient with respect to r is

$$\nabla_r \text{MSBE} = \mathbb{E} [d(\phi' - \phi)],$$

where the derivative with respect to s_i , $i = 1, \dots, K_s$, is

$$\frac{\partial \text{MSBE}}{\partial s_i} = \mathbb{E} \left[d \left(\frac{\partial \phi'^\top}{\partial s_i} - \frac{\partial \phi^\top}{\partial s_i} \right) r \right].$$

Based on this we have the following SA algorithm, that is similar to Algorithm 7 except for the iterates for r_n and s_n .

Algorithm 8. - *Adaptive Basis for Bellman Error (ABBE).* Consider the iterates for η and θ in Algorithm 7. The iterates for r and s_i are

$$\begin{aligned} r_{n+1} &= r_n - \alpha_n^{(3)} d_n (\phi'_n - \phi_n), \\ s_{i,n+1} &= H_P^{(s)} \left[s_{i,n} - \alpha_n^{(1)} d_n \left(\frac{\partial \phi'_n}{\partial s_i} - \frac{\partial \phi_n}{\partial s_i} \right)^\top r_n \right], \quad i = 1, \dots, K_s. \end{aligned}$$

3.4 Minimum Square Projected Bellman Error

The Minimum Square Projected Bellman Error (MSPBE) is defined as

$$\text{MSPBE} = \mathbb{E} \left[\left(\Pi T\tilde{J}(x) - \tilde{J}(x) \right)^2 \right] = \mathbb{E} [d\phi]' (\mathbb{E} [\phi\phi'])^{-1} \mathbb{E} [d\phi],$$

where the projection operator is defined in (6) and where the second equality was proved by Sutton et al. [17], Section 4. We note that the projection operator is independent of r but depend on the basis parameter s . Define $w = (\mathbb{E} [\phi\phi'])^{-1} \mathbb{E} [d\phi]$. Thus, w is the solution to the equation $(\mathbb{E} [\phi\phi']) w = \mathbb{E} [d\phi]$, which yields $\text{MSPBE} = w' \mathbb{E} [d\phi]$. Define similar to [4] section 6.3.3 $Ar + b \triangleq \mathbb{E} [d\phi]$, where $A = \mathbb{E} [\phi(\phi' - \phi)^\top]$ and $b = \mathbb{E} [\phi(g - \eta)]$. Define $A^{(i)}$ to be the i -th column of A . For later use, we give here the gradient of w with respect to r and s in implicit form

$$\begin{aligned} (\mathbb{E} [\phi\phi^\top]) \frac{\partial}{\partial r_i} w &= A^{(i)}, \\ \mathbb{E} [\phi\phi^\top] \frac{\partial}{\partial s_i} w + \frac{\partial}{\partial s_i} \mathbb{E} [\phi\phi^\top] w &= \frac{\partial A}{\partial s_i} r + \frac{\partial b}{\partial s_i}. \end{aligned}$$

Denote by $A_n, A_{i,n}^s, b_{i,n}^s, w_n, w_{i,n}^r$, and $w_{i,n}^s$ the estimators at time n of $A, \partial A/\partial s_i, \partial b/\partial s_i, w, \partial w/\partial r_i$, and $\partial w/\partial s_i$, respectively. Define $A_n^{(i)}$ to be the i -th column of A_n . Thus, the SA iterations for these estimators are

$$\begin{aligned} A_{n+1} &= A_n + \alpha_n^{(4)} \left(\phi_n (\phi_n - \phi_{n+1})^\top - A_n \right), \\ A_{i,n+1}^s &= A_{i,n}^s + \alpha_n^{(4)} \left(\frac{\partial \phi_n}{\partial s_i} (\phi_n - \phi_{n+1})^\top + \phi_n \frac{\partial}{\partial s_i} (\phi_n - \phi_{n+1})^\top - A_{i,n}^s \right), \\ b_{i,n+1}^s &= b_{i,n}^s + \alpha_n^{(4)} \left(g \frac{\partial \phi_n}{\partial s_i} - b_{i,n}^s \right), \\ w_{n+1} &= w_n + \alpha_n^{(4)} (\phi_n d_n - \phi_n \phi_n^\top w_n), \\ w_{i,n+1}^r &= w_{i,n}^r + \alpha_n^{(4)} \left(A_n^{(i)} - \phi_n \phi_n^\top w_{i,n}^r \right), \\ w_{i,n+1}^s &= w_{i,n}^s + \alpha_n^{(4)} \left(A_{i,n}^s r_n + b_{i,n}^s - \left(\frac{\partial}{\partial s_i} (\phi_n \phi_n^\top) \right) w_n - \phi_n \phi_n^\top w_{i,n}^s \right). \end{aligned}$$

where $\{\alpha_n^{(4)}\}$ satisfies Assumption 3. Next, we compute the gradient of the objective function MSPBE with respect to r and s and suggest a gradient descent algorithm to find the optimal value. Thus,

$$\begin{aligned} \frac{\partial \text{MSPBE}}{\partial r_i} &= \mathbb{E}[d\phi]^\top \frac{\partial}{\partial r_i} w^\top + w^\top \frac{\partial}{\partial r_i} \mathbb{E}[d\phi], \\ \frac{\partial \text{MSPBE}}{\partial s_i} &= \frac{\partial w^\top}{\partial s_i} \mathbb{E}[d\phi] + w^\top \frac{\partial \mathbb{E}[d\phi]}{\partial s_i}. \end{aligned}$$

The following algorithm gives the SA iterates for r and s , where the iterates for η and θ are the same as in Algorithms 7 and 8 and therefore omitted. This algorithm has four time scales. The fastest time scale, related to the step sizes $\{\alpha_n^{(4)}\}$, is the estimators time scale, i.e., the estimators for $A, \partial A/\partial s_i, \partial b/\partial s_i, w, \partial w/\partial r_i$, and $\partial w/\partial s_i$. The linear parameters of the critic, i.e., r and η , related to the step sizes $\{\alpha_n^{(3)}\}$, estimated on the second fastest time scale. The actor parameter θ , related to the step sizes $\{\alpha_n^{(2)}\}$, is estimated on the second slowest time scale. Finally, the critic non-linear parameter s , related to the step sizes $\{\alpha_n^{(1)}\}$, is estimated on the slowest time scale. We note that a version where the two fastest times scales operate on a joint single fastest time scale is possible, but results additional technical difficulties in the convergence proof.

Algorithm 9. - *Adaptive Basis for PBE (ABPBE).* Consider the iterates for η and θ in Algorithm 7. The iterates for r and s are

$$\begin{aligned} r_{i,n+1} &= r_{i,n} - \alpha_n^{(3)} \left(d_n \phi_n^\top w_{i,n}^r + w_n^\top A_n^{(i)} r_{i,n} r_n \right), \\ s_{i,n+1} &= s_{i,n} - \alpha_n^{(1)} \left(d_n \phi_n^\top w_{i,n}^s + \left(A_{i,n}^s r_n + b_{i,n}^s \right)^\top w_n \right), \quad i = 1, \dots, K_s. \end{aligned}$$

4 Analysis

In this section we prove the convergence of the previous section Algorithm 7 and 8. We omit the convergence proof of Algorithm 9 that is similar to the convergence proof of Algorithm 8.

4.1 Convergence of ABTD

We begin by stating a theorem regarding the ABTD convergence. Due to space limitations, we give only a proof sketch based on the convergence proof of Theorem 2 of Bhatnagar et al. [5]. The self-contained proof under more general conditions is left to the long version of this work.

Theorem 10. *Consider Algorithm 7 and suppose Assumption 1, 3, and 6, hold. Then, the iterates (17)-(20) of Algorithm 7 converge w.p. 1 to a point that locally maximizes η and solves the equation $E[d\nabla_s \phi^\top r] = 0$.*

Proof. (Sketch) There are three time-scales in (17)-(20), therefore, we wish to use Theorem 5, i.e., we need to prove that the requirements of Assumption 4 are valid w.r.t. to all iterations, i.e., η_n , r_n , θ_n , and s_n .

Requirement 1-4 w.r.t. iterates η_n , r_n , θ_n . Bhatnagar et al. proved in [5] that (17)-(19) converge for a specific s . Assumption 6 implies that the requirements 1-4 of Assumption 4 are valid regarding the iterates of η_n , r_n and θ_n uniformly for all $s \in \mathbb{R}^{K_s}$. Therefore, it sufficient to prove that on top of (17)-(19) also iterate (20) converges, i.e., that requirements 1-4 of Assumption 4 are valid w.r.t. s_n .

Requirement 1 w.r.t. iterate s_n . Define the σ -algebra $\mathcal{F}_n \triangleq \sigma(\eta_k, r_k, \theta_k, s_k : k \leq n)$, and define $F_n^{(\eta)} \triangleq E[g_n - \eta_n | \mathcal{F}_n]$, $F_n^{(r)} \triangleq E[d_n \phi_n | \mathcal{F}_n]$, $F_n^{(\theta)} \triangleq H_P^{(\theta)} E[\psi_n d_n | \mathcal{F}_n]$, $F_n^{(s_i)} \triangleq H_P^{(s)} E[d_n \frac{\partial \phi_n^\top}{\partial s_i} r_n | \mathcal{F}_n]$, and $M_{n+1}^{(s_i)} \triangleq H_P^{(s)} [(d_n \frac{\partial \phi_n^\top}{\partial s_i} r_n) - F_n^{(s_i)}]$. Thus, (20) can be expressed as

$$s_{i,n+1} = s_{i,n} + \alpha_n^{(1)} \left(F_n^{(s_i)} + M_{n+1}^{(s_i)} \right). \quad (21)$$

Trivially, using Assumption 6, $F_n^{(r)}$, $F_n^{(\theta)}$, and $F_n^{(s)}$ are Lipschitz, with respect to s , with coefficients B_ϕ^2 , L_ϕ , and L_ϕ , respectively. Also, $F_n^{(s_i)}$ is Lipschitz with respect to η , r , and θ with coefficients 1, B_ϕ , and 1, respectively. Thus, requirement 1 of Assumption 4 is valid.

Requirements 2 and 3 w.r.t. iterate s_n . By construction, the iterate s_n is bounded. Requirement 3 of Assumption 4 is valid using the boundedness of the martingale difference noise $M_{n+1}^{(s_i)}$ that implies, using the martingale convergence theorem [4], that the martingale $\sum_n \alpha_n^{(3)} M_{n+1}^{(s_i)}$ converges.

Requirement 4 w.r.t. iterate s_n . Using the result of Bhatnagar et al. [5], the fast time scales converge w.r.t. the slow time scale. Thus, Requirement 4 is valid based on the fact that the iterates (17)-(19) converge. \square

4.2 Convergence of Adaptive Basis for Bellman Error

We begin by stating the theorem and then we prove it.

Theorem 11. *Consider Algorithm 8 and suppose that Assumption 1, 3, and 6, hold. Then, Algorithm 8 converge w.p. 1 to a point that locally maximizes η and locally minimizes $E[d^2]$.*

Proof. (Sketch) To use Theorem 5 we need to check that Assumption 4 is valid. Define the σ -algebra $\mathcal{F}_n \triangleq \sigma(\eta_k, r_k, \theta_k, s_k : k \leq n)$, and define $F_n^{(\eta)} \triangleq E[g_n - \eta_n | \mathcal{F}_n]$, $M_{n+1}^{(\eta)} \triangleq (g_n - \eta_n) - F_n^{(\eta)}$, $F_n^{(r)} \triangleq -E[d_n(\phi_{n+1} - \phi_n) | \mathcal{F}_n]$, $M_{n+1}^{(r)} \triangleq -(d_n(\phi_{n+1} - \phi_n)) - F_n^{(r)}$, $F_n^{(\theta)} \triangleq E[\psi_n d_n | \mathcal{F}_n]$, $M_{n+1}^{(\theta)} \triangleq (\psi_n d_n) - F_n^{(\theta)}$, $F_n^{(s_i)} \triangleq -E[d_n(\frac{\partial \phi_{n+1}^\top}{\partial s_i} r_n - \frac{\partial \phi_n^\top}{\partial s_i} r_n) | \mathcal{F}_n]$, and $M_{n+1}^{(s_i)} \triangleq -(d_n(\frac{\partial \phi_{n+1}^\top}{\partial s_i} r_n) - \frac{\partial \phi_n^\top}{\partial s_i} r_n) - F_n^{(s_i)}$.

On the fast time scale (which is related to $a_n^{(3)}$), as in Theorem 10, η_n converges to $E[g(x)]$. On the same time scale we need to show that the iterate for r_n converges. Using the above definitions, we can write the iteration r_n as

$$r_{n+1} = r_n + \alpha_n^{(3)} \left(F_n^{(r)} + M_{n+1}^{(r)} \right). \quad (22)$$

We use Theorem 2.2 of Borkar and Meyn [7] to achieve this. Briefly, this theorem states that given an iteration as (22), this iteration is bounded w.p.1 if

- (A1) The process $F_n^{(r)}$ is Lipschitz, the function $F_\infty(\sigma) \triangleq \lim_{\sigma \rightarrow \infty} F^{(r)}(\sigma r)/r$ is Lipschitz, and $F_\infty(\sigma)$ is asymptotically stable in the origin.
- (A2) The sequence $M_{n+1}^{(r)}$ is a martingale difference noise and for some C_0

$$E \left[(M_{n+1}^{(r)})^2 | \mathcal{F}_n \right] \leq C_0(1 + \|r_n\|^2).$$

Trivially, the function $F_n^{(r)}$ is Lipschitz continuous, and we have

$$\lim_{\sigma \rightarrow \infty} F^{(r)}(\sigma r)/r = -E[(\phi' - \phi)(\phi' - \phi)^\top] r.$$

Thus, it is easy to show, using Assumption 6, that the ODE $\dot{r} = F_\infty^{(r)}$ has a unique global asymptotically stable point at the origin and (A1) is valid. For (A2) we have

$$\begin{aligned} E \left[\left\| M(n+1)^{(r)} \right\|^2 \middle| \mathcal{F}_n \right] &\leq E \left[\|d_n(\phi'_n - \phi_n)\|^2 \middle| \mathcal{F}_n \right] \\ &\leq 2(B_g + B_\eta + 4B_\phi^2 r_n)^2 \triangleq K''(1 + \|r_n\|^2), \end{aligned}$$

where the first inequality results from the inequality $E[(x - E[x])^2] \leq E[x^2]$, and the second inequality results from the uniform boundedness of the involved variables. We note that the related ODE for this iteration is given by $\dot{r} = F^{(r)}$, and the related Lyapunov function is given by $E[d^2]$. Next, we need show that under the convergence of the fast time scales for η_n and r_n , the slower iterate

for θ converges. The proof of this is identical to that of Theorem 2 of [5] and is therefore omitted. We are left with proving that if the fast timescales converge, i.e., the iterates η_n , r_n , and θ_n , then the iterate $s_n^{(i)}$ converge as well. The proof follows similar lines as of the proof for $s_n^{(i)}$ in the proof of Theorem 10, whereas here the iterate s_n converge to the stable point of the ODE $\dot{s} = \nabla_s E[d(x, y)^2]$. \square

5 Simulations

In this section we report empirical results applying the algorithms on two types of problems: GARNET problems [1] and the mountain car problem.

5.1 Garnet problems

The GARNET² problems [1,5] are a class of randomly constructed finite MDPs serving as a test-bench for RL algorithms. A GARNET problem is characterized by four parameters and is denoted by $\text{GARNET}(X, U, B, \sigma)$. The parameter X is the number of states, U is the number of actions, B is the branching factor, and σ is the variance of each transition reward. When constructing such a problem, we generate for each state a reward, distributed according to $\mathcal{N}(0, 1)$. For each state-action the reward is distributed according to $\mathcal{N}(g(x), \sigma^2)$. The transition matrix for each action is composed of B non-zero terms. We consider the same GARNET problems as those simulated by [5]. For the critic's feature vector, we use the basis functions $\phi(x, s) = \cos\left(\frac{x}{d}s + \varrho_{x,d}\right)$, where $x = 1, \dots, N$, $1 \leq d \leq K_r$, $s \in \mathbb{R}^1$, and $\varrho_{x,d}$ are i.i.d. uniform random phases. Note that only one parameter in this simulation controls the basis functions. The actor's feature vectors are of size $K_a \times |U|$, and are constructed as

$$\xi(x, u) \triangleq (\underbrace{0, \dots, 0}_{K_a \times (u-1)}, \phi(x, s(t=0)), \underbrace{0, \dots, 0}_{K_a \times (|U|-u)}).$$

The policy function is $\mu(u|x, \theta) = e^{\theta^\top \xi(x, u)} / \sum_{u' \in U} e^{\theta^\top \xi(x, u')}$. Bhatnagar et al. [5] reported simulation results for two GARNET problems: GARNET(30, 4, 2, 0.1) and GARNET(100, 10, 3, 0.1). We based our simulations on these results where the time steps are identical to those of [5]. The GARNET(30, 4, 2, 0.1) problem (Fig. 1 left pane) was simulated for $K_r = 4$ (two lower graphs) and $K_r = 12$ (two upper graphs), where each graph is an average of 100 repeats. The GARNET(100, 10, 3, 0.1) problem (Fig. 1 right pane) was simulated for $K_r = 4$ (two lower graphs) and $K_r = 12$ (two upper graphs), where each graph is an average of 100 repeats. We can see that in such problems there is an evident advantage to an adaptive base, which can achieve additional fitness to the problem, and thus even for low dimensional problems the adaptation may be crucial.

5.2 The Mountain Car

The mountain car task (see [15] or [16] for details) is a physical problem where a car is positioned randomly between two mountains (see Fig. 2 left pane) and

² brevity for Generic Average Reward Non-stationary Environment Test-bench

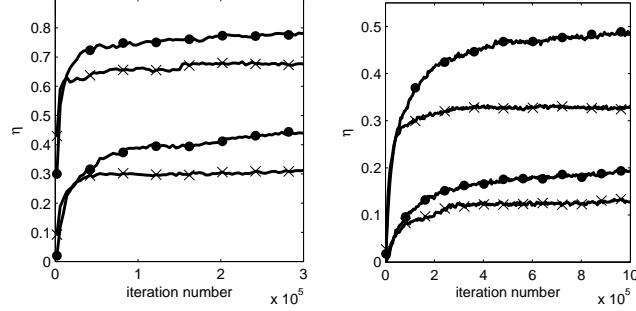


Fig. 1. Results for GARNET(30, 4, 2, 0.1) (left pane) and GARNET(100, 10, 3, 0.1) (right pane) where circled graphs are for adaptive bases. In each graph the lower two graphs are for $K_r = 4$ and the upper graphs are for $K_r = 12$. See text for detail.

needs to climb the right mountain, but the engine of the car does not support such a straight climb. Thus, the car needs to accumulate sufficient gradational energy, by applying back and forth actions, in order to succeed.

We applied the adaptive basis TD algorithm on this problem. We chose the critic basis functions to be radial basis functions (RBF) (see [8]), where the value function is represented by $\sum_{i=1}^M r_i \exp\{-(p - s_i^{(p)})^2/s_{p,i}^2 - (v - s_i^{(v)})^2/s_{v,i}^2\}$. The centers of the RBFs are parameterized by $(s_i^{(p)}, s_i^{(v)})_{i=1}^M$ while the variance is represented by $(s_{p,i}^2, s_{v,i}^2)_{i=1}^M$. In the right pane of Fig. 2 we present simulation results for 4 cases: SARSA (blue dash) which is based on the implementation of [15], AC (red dash-dot) with 64 basis functions uniformly distributed on the parameter space, ABTD with 64 basis functions (magenta dotted) where both the location and the variance of the basis functions can adapt, ABAC with 16 basis functions (black solid) with the same adaptation. We see that the adaptive basis gives a significant advantage in performance. Moreover, we see that even with small number of parameters, the performance is not affected. In the middle pane, the dynamics of a realization of the basis functions is presented where the dots and circles are the initial positions and final positions of the basis functions, respectively. The circle sizes are proportional to the basis functions standard deviations, i.e., $(s_{p,i}, s_{v,i})_{i=1}^M$.

5.3 The Performance of Multiple Time Scales vs. Single Time Scale

In this section we discuss the differences in performance between the MTS algorithm to the STS algorithms. Unlike mistakenly thought, neither MTS algorithms nor STS algorithms have advantage in terms of convergence. This difference comes from the fact that both methods perform the gradient algorithm differently, thus, they may result different trajectories. In Fig. 3 we can see a case on a GARNET(30, 5, 5, 0.1) where the MTS ABTD algorithm (upper red diamond graph) has an advantage over STS ABTD algorithms or MTS static basis AC algorithm as in [5] (rest of the graphs). We note that this is not always the case and it depends on the problem parameters or the initial conditions.

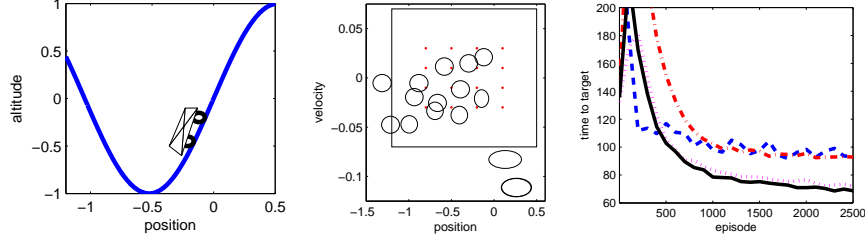


Fig. 2. (left pane) illustration of the mountain car task. (middle pane) Realization of ABTD with 16 basis functions where the red dots are the basis functions initial position and the circles are their final position. The radii are proportional to the variance. The rectangle represents the bounded parameter set of the car. (right pane) Simulation result for the mountain car problem with solutions of SARSA (blue dash) AC (red dash-dot) AB-AC with 64 basis functions (magenta dotted) AB-AC with 16 basis functions (black solid).

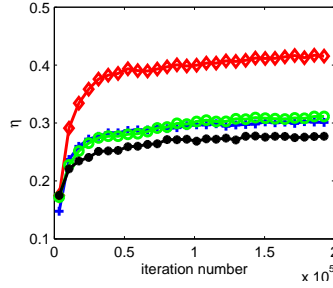


Fig. 3. Results for GARNET(30, 5, 5, 0.1) for $K_r = 8$. The upper diamond red graph is MTS ABTD algorithm, the circled green graph is STS ABTD acting on slow time scale, the blue crossed line is MTS static basis AC algorithm as in [5], and the black starred line is STS ABTD acting on fast time scale. Each graph is average of 100 simulation runnings.

6 Discussion

We introduced three new AC based algorithms where the critic's basis is adaptive. Convergence proofs, in the average reward case, were provided. We note that the algorithms can be easily transformed to discounted reward. When considering other target functions, more AC algorithms with adaptive basis can be devised, e.g., considering the objective function $\|E[d\phi]\|^2$ yields $A^\top TD$ and GTD(0) algorithms [18]. Also, mixing the different algorithm introduced in here, can yield new algorithms with some desired properties. For example. we can devise an algorithm where the linear part is updated similar to (18) and the non-linear part is updated similar to (21). Convergence of such algorithms will follow the same lines of proof as introduced here.

The advantage of adaptive bases is evident: they relieve the domain expert from the task of carefully designing the basis. Instead, he may choose a flexible basis, where one use algorithms as introduced here to adapt the basis to the problem at hand. From a methodological point of view, the method we introduced in this paper demonstrates how to easily transform an existing RL algorithm to an adaptive basis algorithm. The analysis of the original problem is used to show convergence of the faster time scale and the slow time scale is used for modifying the basis, analogously to “code reuse” concept in software engineering.

References

1. Archibald, T., McKinnon, K., and Thomas, L.: (1995) On the Generation of Markov Decision Processes. *Journal of the Operational Research Society*, **46** (1995) 354-361
2. Bradtke, S. J., Barto, A. G.: Linear least-squares algorithms for temporal difference learning. *Machine Learning*, **22** (1996) 33-57
3. Bertsekas, D.: *Dynamic programming and optimal control*, 3rd ed. Athena Scientific (2007)
4. Bertsekas, D., Tsitsiklis, J.: *Neuro-dynamic programming*. Athena Scientific (1996)
5. Bhatnagar, S., Sutton, R., Ghavamzadeh, M., Lee, M.: Natural actor-critic algorithms. Technical report Univ. of Alberta (2007)
6. Borkar, V.: Stochastic approximation with two time scales. *Systems & Control Letters* **29** 291-294 (1997)
7. Borkar, V., Meyn, S.: The ode method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Cont. and Optim.* **38** (2000) 447-469
8. Haykin, S.: *Neural networks: a comprehensive foundation*. Prentice Hall (2008)
9. Kushner, H., Yin, G.: *Stochastic approximation and recursive algorithms and applications*. Springer Verlag (2003)
10. Leslie, D., Collins, E.: Convergent multiple-timescales reinforcement learning algorithms in normal form games. *The Annals of App. Prob.* **13** (2003) 1231-1251.
11. Menache, I., Mannor, S., Shimkin, N.: Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research* **134** (2006) 215-238
12. Mokkadem, A., Pelletier, M.: Convergence rate and averaging of nonlinear two-time-scale stochastic approximation algorithms. *Annals of Applied Prob.* **16** 1671
13. Polyak, B.: New method of stochastic approximation type. *Automat. Remote Control* **51** (1990) 937-946
14. Puterman, M.: *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons Inc (1994)
15. Singh, S., Sutton, R.: Reinforcement learning with replacing eligibility traces. *Machine learning*, **22** (1996) 123-158.
16. Sutton, R. S., Barto, A. G.: *Reinforcement Learning - an Introduction*. MIT Press, Cambridge, MA, 1998
17. Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., Wiewiora, E.: Fast gradient-descent methods for temporal-difference learning with linear function approximation. *Proceedings of the 26th Annual International Conference on Machine Learning* (2009)
18. Sutton, R. S., Szepesvari, C., Maei, H. R.: A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. *Advances in Neural Information Processing Systems* 21 (2009b) 1609-1616

19. Yu, H., & Bertsekas, D.: Basis function adaptation methods for cost approximation in MDP. Proc. of IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning, Nashville, TN (2009)