IEEE Control Systems Letters paper presented at
2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Actor-Critic or Critic-Actor? A Tale of Two Time Scales

Shalabh Bhatnagar[1], Vivek S. Borkar[2], and Soumyajit Guin[1]

***Abstract*— We revisit the standard formulation of tabular actor-critic algorithm as a two time-scale stochastic approximation with value function computed on a faster time-scale and policy computed on a slower time-scale. This emulates policy iteration. We observe that reversal of the time scales will in fact emulate value iteration and is a legitimate algorithm. We provide a proof of convergence and compare the two empirically with and without function approximation (with both linear and nonlinear function approximators) and observe that our proposed critic-actor algorithm performs on par with actor-critic in terms of both accuracy and computational effort.**

Keywords: Reinforcement learning; approximate dynamic programming; critic-actor algorithm; two time-scale stochastic approximation.

I. Introduction

The actor-critic algorithm of Barto et al. [1] is one of the foremost reinforcement learning algorithms for data-driven approximate dynamic programming for Markov decision processes. Its rigorous analysis as a two time-scale stochastic approximation was initiated in [9] and [10], first in tabular form, then with linear function approximation, respectively. The algorithm has a faster time scale component for value function evaluation (the 'critic'), with policy evaluation on a slower time scale (the 'actor'). Using two time-scale philosophy, the former sees the latter as quasi-static, i.e., varying slowly enough on the time-scale of the critic that critic can treat the actor's output essentially as a constant. In turn, the actor sees the critic as quasi-equilibrated, i.e., tracking the value function corresponding to the current policy estimate of the actor. Thus the scheme emulates policy iteration, wherein one alternates between value function computation for a fixed policy and policy update based on this value function by minimization of the associated 'Q-value'. Another interpretation is that of a Stackelberg game between the actor and the critic, with the actor leading and being followed by the critic. That is, the actor tunes the policy slowly and the critic reacts to it rapidly, so that the actor can factor in the critic's reaction in her update.

This raises the issue as to what would happen if the time scales of the actor and the critic are reversed, rendering critic the leader. We observe below that in this case, the scheme emulates value iteration, making it a valid reinforcement learning scheme. We call it the 'critic-actor' algorithm. This structure as such is not novel and was explored in the context of Q-learning in [3], [4] with a different motivation. Its incorporation into the actor-critic facilitates a dimensionality reduction and a clean convergence analysis. This is so because we work with value and not Q-value functions.

While there are multiple variations of the actor-critic based on the exact scheme used for policy updates, we stick to one of the three proposed in [9] to make our point. It may be recalled that one of the purported advantages of the actor-critic algorithm is that because of the slow time scale of the policy update which renders it quasi-static, the value function update is 'essentially' a linear operation and therefore one can legitimately use schemes such as TD(λ) [12], [14] that are based on linear function aproximation. This is in contrast with, e.g., Q-learning [15] where the nonlinearity of the iterate is not linear function approximation friendly. This problem returns to actor-critic if we interchange the time scales. Nevertheless, given the current trend towards using neural networks for function approximation, this theoretical advantage is no longer there. In particular, this puts the actor-critic and critic-actor schemes a priori on equal footing as far as nonlinear function approximation is concerned.

II. The Basic Framework

Our Markov decision process (MDP) is a random process $X_n, n \geq 0$, in a finite state space S satisfying, a.s.,

$$P(X_{n+1} = j \mid X_k, A_k, k \leq n) = p(X_n, A_n, j), \ \forall n \geq 0.$$

Here $A_n \in U(X_n)$ is the action at time n when the state is X_n where $U(i) :=$ the finite set of actions admissible in state i. Also, $p(i, a, j)$ denotes the transition probability from state i to j when a feasible action a is chosen in state i. $\varphi := \{A_n\}$ with each A_n as above will be said to be admissible. For simplicity, we take $U(\cdot) \equiv U$. A stationary deterministic policy (SDP) f is one where $A_n = f(X_n)$ for some $f : S \to U$. Similarly, if given X_n, A_n is conditionally independent of $\{X_m, A_m, m < n\}$ and has the same conditional law $\pi : S \to \mathcal{P}(U)$ $\forall n$, then π is called a stationary randomised policy (SRP). Here $\mathcal{P}(U) :=$ the space of probability vectors on U. We shall denote the probability vector $\pi(i)$ as $\pi(i) = (\pi(i, a), a \in U(i))^T$. Let $g : S \times U \times S \to \mathbb{R}$ denote the single-stage cost function and $\gamma \in (0, 1)$ the discount factor.

For a given admissible sequence $\varphi = \{A_n\}$, consider the infinite horizon discounted cost

$$V_\varphi(i) \triangleq E\left[\sum_{n=0}^{\infty} \gamma^n g(X_n, A_n, X_{n+1}) \mid X_0 = i\right], \ i \in S. \tag{1}$$

SB was supported by a J. C. Bose Fellowship, Project No. DFTM/ 02/ 3125/M/04/AIR-04 from DRDO under DIA-RCOE, a project from DST-ICPS, and the RBCCPS, IISc.

VSB was supported by the S. S. Bhatnagar Fellowship from the Council of Scientific and Industrial Research, Government of India.

[1]Department of Computer Science and Automation, Indian Institute of Science, Bengaluru 560012, India shalabh@iisc.ac.in, gsoumyajit@iisc.ac.in

[2]Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai 400076, India borkar.vs@gmail.com

The function $V_\pi := V_\varphi$ for $\varphi \approx$ an SRP π is called the value function under the SRP π. The corresponding (linear) Bellman equation takes the form

$$V_\pi(i) = \sum_{a \in U(i)} \pi(i,a) \sum_{j \in S} p(i,a,j)(g(i,a,j) + \gamma V_\pi(j)). \tag{2}$$

Define the 'value function'

$$V^*(i) = \min_\varphi V_\varphi(i). \tag{3}$$

This satisfies the Bellman equation

$$V^*(i) = \min_{a \in U(i)} \left(\sum_{j \in S} p(i,a,j)(g(i,a,j) + \gamma V^*(j) \right). \tag{4}$$

Define the Q-value function $Q^*(\cdot,\cdot) : S \times U \to \mathcal{R}$ so that $Q^*(i,a) :=$ the total cost incurred if starting in state i, action a is chosen and the optimal action is chosen in each state visited subsequently, i.e.,

$$Q^*(i,a) = \sum_{j \in S} p(i,a,j)(g(i,a,j) + \gamma V^*(j)). \tag{5}$$

Then an SRP π^* would be optimal if support$(\pi^*(i)) \subset \arg\min Q^*(i,\cdot)$. In particular, an optimal SDP (hence SRP) exists and is given by any choice of minimiser of $Q^*(i,\cdot)$ for each i. See [11] for an extensive treatment.

Policy iteration (PI) and value iteration (VI) are two of the numerical procedures for solving the Bellman equation. Whereas actor-critic algorithms [9] emulate PI, the algorithm with the timescales of the actor-critic algorithm reversed will be seen to mimic VI.

III. The Proposed Critic-Actor Algorithm

Let $V_n(\cdot)$ and $\pi_n(\cdot,\cdot)$ denote the estimates at instant n of the value function and the optimal SRP. We consider here an off-policy setting where states and state-action tuples are sampled from a priori given distributions in order to decide the particular state whose value is updated next, as well as the action-probability estimate (in the SRP) of the sampled state-action tuple to be updated. This is more general than the on-policy setting commonly considered in standard actor-critic algorithms such as [10], [5], as the latter turns out to be a special case of the off-policy setting we consider.

Let $\{Y_n\}$ and $\{Z_n\}$ be S and $S \times U$ valued processes such that if $Y_n = i \in S$, then the value of state i, denoted by $V_n(i)$, is updated at the nth iterate. Likewise if $Z_n = (i,a)$, then the policy component corresponding to the (i,a)-tuple is updated. Define $\{\nu_1(i,n)\}, \{\nu_2(i,a,n)\}$ by (for $n > 0$):

$$\nu_1(i,n) = \sum_{m=0}^{n-1} I\{Y_m = i\}, \text{ with } \nu_1(i,0) = 0,$$

$$\nu_2(i,a,n) = \sum_{m=0}^{n-1} I\{Z_m = (i,a)\}, \text{ with } \nu_2(i,a,0) = 0.$$

As with policy gradient schemes [13], we parameterize the policy. Specifically we consider a parameterised Boltzmann or Gibbs form for the SRP as below.

$$\pi_\theta(i,a) = \frac{\exp(\theta(i,a))}{\sum_b \exp(\theta(i,b))}, \ i \in S, a \in U. \tag{6}$$

For a given $\theta_0 \gg 0$, let $\Gamma_{\theta_0} : \mathbb{R} \to [-\theta_0, \theta_0]$ denote the projection map. Let $\{a(n)\}, \{b(n)\}$ be positive step size sequences satisfying conditions we specify later.

The Critic-Actor Algorithm

The proposed critic-actor algorithm has similar set of updates as Algorithm 3 of [9] but with the timescales reversed, i.e., $a(n) = o(b(n))$. Let $\{\xi_n(i,a)\}$ and $\{\eta_n(i,a)\}$, $i \in S$, $a \in U$ be independent families of i.i.d random variables with law $p(i,a,\cdot)$. Let $\{\phi_n(i)\}$ be a sequence of U-valued i.i.d random variables with the conditional law of $\phi_n(i)$ given the sigma-algebra $\sigma(V_m(\cdot), \pi_m(\cdot), \xi_m(\cdot,\cdot), \eta_m(\cdot,\cdot), m \le n)$ generated by the random variables realised till n, being denoted by $\pi_n(i,\cdot)$. The critic and actor recursions now take the following form:

$$\begin{aligned} V_{n+1}(i) = V_n(i) + a(\nu_1(i,n))[&g(i,\phi_n(i),\xi_n(i,\phi_n(i))) \\ &+ \gamma V_n(\xi_n(i,\phi_n(i))) - V_n(i)] I\{Y_n = i\}, \end{aligned} \tag{7}$$

$$\begin{aligned} \theta_{n+1}(i,a) = \Gamma_{\theta_0}(&\theta_n(i,a) + b(\nu_2(i,a,n))[V_n(i) \\ &- g(i,a,\eta_n(i,a)) - \gamma V_n(\eta_n(i,a))] I\{Z_n = (i,a)\}). \end{aligned} \tag{8}$$

The V_n update is governed by the step-size sequence $a(n), n \ge 0$, while the θ_n update is governed by $b(n), n \ge 0$. From Assumption 1 below, this implies that the V_n update proceeds on a slower timescale in comparison to the θ_n update. In the next section, we give a proof of convergence of our critic-actor algorithm.

IV. Convergence of Critic-Actor Scheme

For $x > 0$, define

$$\begin{aligned} N_1(n,x) &= \min\{m > n \mid \sum_{i=n+1}^{m} \bar{a}(i) \ge x\}, \\ N_2(n,x) &= \min\{m > n \mid \sum_{i=n+1}^{m} \bar{b}(i) \ge x\}, \end{aligned}$$

where $\bar{a}(n) \equiv a(\nu_1(i,n))$ and $\bar{b}(n) \equiv b(\nu_2(i,a,n))$, respectively. We make the following assumptions.

Assumption 1 (Step-Sizes): $a(n), b(n), n \ge 0$ are eventually non-increasing and satisfy the following conditions:

(i) $\sum_n a(n) = \sum_n b(n) = \infty$.

(ii) $\sum_n (a(n)^2 + b(n)^2) < \infty$.

(iii) $a(n) = o(b(n))$.

(iv) For any $x \in (0,1)$, $\sup_n \frac{a([xn])}{a(n)}, \sup_n \frac{b([xn])}{b(n)} < \infty$, where $[xn]$ denotes the integer part of xn.

(v) For any $x \in (0,1)$, $A(n) := \sum_{i=0}^{n} a(i)$, $B(n) := \sum_{i=0}^{n} b(i)$, $n \ge 0$, we have $\frac{A([yn])}{A(n)}, \frac{B([yn])}{B(n)} \to 1$, as $n \to \infty$, uniformly over $y \in [x,1]$.

Assumption 2 (Frequent Updates): (i) There exists a constant $\kappa > 0$ such that the following conditions hold almost surely: $\liminf_{n\to\infty} \frac{\nu_1(i,n)}{n} \geq \kappa,\ \forall i \in S$, $\liminf_{n\to\infty} \frac{\nu_2(i,a,n)}{n} \geq \kappa,\ \forall (i,a) \in S \times U$.

(ii) For $x > 0$, the limits

$$\lim_{n\to\infty} \frac{\sum_{j=\nu_1(i,n)}^{\nu_1(i,N_1(n,x))} a(j)}{\sum_{j=\nu_1(k,n)}^{\nu_1(k,N_1(n,x))} a(j)}, \quad \lim_{n\to\infty} \frac{\sum_{j=\nu_2(i,a,N_2(n,x))}^{\nu_2(i,a,N_2(n,x))} b(j)}{\sum_{j=\nu_2(k,b,n)}^{\nu_2(k,b,N_2(n,x))} b(j)},$$

exist almost surely for states i, k and state-action tuples (i,a) and (k,b) in the two limits respectively.

Since $\nu_1(i,n) \leq n, \forall i$, and $\nu_2(i,a,n) \leq n, \forall (i,a)$ tuples, we have, for n sufficiently large, $a(n) \leq \bar{a}(n)$, and $b(n) \leq \bar{b}(n)$, respectively. Assumption 1(i)-(ii) are the standard Robbins-Monro conditions for stochastic approximation algorithms (see Ch. 2 of [8]). Condition (iii) is standard for two time-scale stochastic approximations (Section 8.1, [8]). Conditions (iv), (v) and Assumption 2 are required for handling the asynchronous nature of the iterates (see Ch. 6 of [8], also [7]). Examples of step-sizes that satisfy Assumptions 1 and 2(ii) include (see [9]) (i) $a(n) = 1/(n+1), b(n) = \log(n+2)/(n+2)$, (ii) $a(n) = 1/((n+2)\log(n+2))$, $b(n) = 1/(n+1)$, etc., for $n \geq 0$. Assumption 2(i) is satisfied for instance if $\{Z_n\}$ is ergodic and $\{Y_n\}$ is ergodic under any given policy dictated by $\{\theta_n\}$. If these are independently sampled from some distributions, the same should assign positive probabilities to all state and state-action components.

A. Convergence of the Faster Recursion

It has been shown in Lemma 4.6 of [9] that $\bar{a}(n), \bar{b}(n) \to 0$ as $n \to \infty$. Further, $\bar{a}(n) = o(\bar{b}(n))$. Since our V-update proceeds on the slower timescale, we let $V_n \approx V$ for the analysis of the faster recursion. For all (i,a), let

$$\begin{aligned} g_{ia}(V) &= V(i) - \sum_j p(i,a,j)(g(i,a,j) + \gamma V(j)), \\ k_{ia}(V) &= \sum_j p(i,a,j)(g(i,a,j) + \gamma V(j)) - V(i). \end{aligned}$$

Let $\mathcal{F}_n \stackrel{\triangle}{=} \sigma(\xi_m(i,a), \eta_m(i,a), \phi_m(i), m < n;\ Y_m, Z_m, V_m, \theta_m(i,a),\ m \leq n, i \in S, a \in U), n \geq 0$, denote an increasing family of associated sigma fields. Let

$$\begin{aligned} &\mu_n^2(i,a) = I\{Z_n = (i,a)\},\ \forall (i,a) \in S \times U, \\ &\mu_n^1(i) = I\{Y_n = i\},\ \forall i \in S, \\ &\mathcal{M}^{\mu_n^2} = \text{diag}(\mu_n^2(i,a), (i,a) \in S \times U), \\ &\mathcal{M}^{\mu_n^1} = \text{diag}(\mu_n^1(i), i \in S). \end{aligned}$$

For $n \geq 0$, define the $\{\mathcal{F}_n\}$-adapted sequences

$$\begin{aligned} M_n(i,a) &= V(i) - g(i,a,\eta_{n-1}(i,a)) - \gamma V(\eta_{n-1}(i,a)) \\ &\quad + k_{ia}(V), \\ N_n(i) &= g(i,\phi_{n-1}(i),\xi_{n-1}(i,\phi_{n-1}(i))) - V(i) \\ &\quad + \gamma V(\xi_{n-1}(i,\phi_{n-1}(i))) - \sum_{a\in U} \pi_{\theta_n}(i,a) k_{ia}(V). \end{aligned}$$

Lemma 1: The sequences

$$\sum_{n=1}^{m} b(\nu_2(i,a,n)) M_{n+1}(i,a) I\{Z_n = (i,a)\},\ m \geq 1, \text{ and}$$

$$\sum_{n=1}^{m} a(\nu_1(i,n)) N_{n+1}(i) I\{Y_n = i\},\ m \geq 1,$$

converge almost surely as $m \to \infty$.

Proof: Follows as in Lemma 4.5 of [9]. ■

The analysis of (7) proceeds by rewriting it as:

$$\begin{aligned} \theta_{n+1}(i,a) &= \Gamma_{\theta_0}(\theta_n(i,a) + \bar{b}(n)\mathcal{M}^{\mu_n^2}[V(i) \\ &\quad - g(i,a,\eta_n(i,a)) - \gamma V(\eta_n(i,a))]). \end{aligned}$$

One may further rewrite it as follows:

$$\theta_{n+1}(i,a) = \Gamma_{\theta_0}(\theta_n(i,a) + \bar{b}(n)\mathcal{M}^{\mu_n^2}[g_{ia}(V) + M_{n+1}(i,a)]).$$

Let

$$\bar{\Gamma}_{\theta_0}(g_{ia}(V)) = \lim_{\Delta\to 0} \left(\frac{\Gamma_{\theta_0}(\theta(i,a) + \Delta g_{ia}(V)) - \theta(i,a)}{\Delta} \right).$$

Consider now the ODE

$$\dot{\theta}(i,a) = \bar{\Gamma}_{\theta_0}(g_{ia}(V)). \tag{9}$$

Let

$$\gamma_{ia}(\theta) = \begin{cases} 0 & \text{if } \theta(i,a) = \theta_0 \text{ and } k_{ia}(V) \leq 0, \\ & \text{or } \theta(i,a) = -\theta_0 \text{ and } k_{ia}(V) \geq 0, \\ 1 & \text{otherwise.} \end{cases}$$

Then it can be seen (see Sec.5.4 of [9]) that

$$\bar{\Gamma}_{\theta_0}(g_{ia}(V)) = -k_{ia}(V)\gamma_{ia}(\theta).$$

Thus, the ODE (9) takes the form:

$$\dot{\theta}(i,a) = -k_{ia}(V)\gamma_{ia}(\theta). \tag{10}$$

From the parameterised form of the policy (6), it follows from (10) that for $(i,a) \in S \times U$,

$$\begin{aligned} \dot{\pi}_\theta(i,a) &= \pi_\theta(i,a)(\dot{\theta}(i,a) - \sum_{b\in U} \pi_\theta(i,b)\dot{\theta}(i,b)), \\ &= \pi_\theta(i,a)(\bar{\Gamma}_{\theta_0}(g_{ia}(V)) - \sum_{b\in U} \pi_\theta(i,b)\bar{\Gamma}_{\theta_0}(g_{ib}(V))), \\ &= -\pi_\theta(i,a)(k_{ia}(V)\gamma_{ia}(\theta) - \sum_{b\in U} \pi_\theta(i,b)k_{ib}(V)\gamma_{ib}(\theta)). \end{aligned}$$

This is a replicator dynamics whose stable attractors are sets consisting of the π_θ that minimize $\sum_{(i,a)\in S\times U} k_{ia}(V)$, i.e., θ that satisfy $\theta(i,a) = \theta_0$ for at least one $a \in \arg\min(k_{i\cdot}(V))$ and $\theta(i,a) = -\theta_0$ otherwise (see Lemma 5.10 of [9])[1]. Let $\hat{\pi}$ denote one such policy and $\hat{V}$ the corresponding value function. Then

$$\hat{V}(i) = \sum_{a\in U} \hat{\pi}(i,a) \sum_j p(i,a,j)(g(i,a,j) + \gamma \hat{V}(j)),\ \forall i.$$

[1]Under reasonable conditions on noise, it is known that stochastic approximation converges a.s. to its stable attractors, see, e.g., section 3.4 of [8] and the references therein.

Define a policy π^* that corresponds to $\theta_0 \to \infty$ in the above. Then this is an optimal policy and the corresponding value function V^* satisfies the corresponding dynamic programming equation.

$$V^*(i) = \sum_{a \in U} \pi^*(i,a) \sum_j p(i,a,j)(g(i,a,j) + \gamma V^*(j)), \ \forall i.$$

Proposition 2: We have

$$\|\hat{V} - V^*\|_\infty \leq \frac{\max_i \|\hat{\pi}(i,\cdot) - \pi^*(i,\cdot)\|_1 \max_i \|\bar{g}(i,\cdot)\|_\infty}{(1-\gamma)^2},$$

where $\bar{g}(i,a) = \sum_j p(i,a,j) g(i,a,j)$.

Proof: Note that

$$|\hat{V}(i) - V^*(i)| \leq \sum_a |\hat{\pi}(i,a) - \pi^*(i,a)||\bar{g}(i,a)| + \gamma \sum_a \hat{\pi}(i,a) \sum_j p(i,a,j)|\hat{V}(j) - V^*(j)| + \gamma \sum_a |\hat{\pi}(i,a) - \pi^*(i,a)| \sum_j p(i,a,j)|V^*(j)|. \quad (11)$$

Thus,

$$\|\hat{V} - V^*\|_\infty \leq \max_i \|\hat{\pi}(i,\cdot) - \pi^*(i,\cdot)\|_1 \max_i \|\bar{g}(i,\cdot)\|_\infty + \gamma \|\hat{V} - V^*\|_\infty + \gamma \max_i \|\hat{\pi}(i,\cdot) - \pi^*(i,\cdot)\|_1 \|V^*\|_\infty. \quad (12)$$

Now note that by definition

$$\|V^*\|_\infty \leq (1-\gamma)^{-1} \max_i \|\bar{g}(i,\cdot)\|_\infty. \quad (13)$$

The claim follows upon substituting (13) in (12). ■

Theorem 3: Given $\epsilon > 0$, there exists $\bar{\theta} > 0$ such that $\forall\ \theta_0 > \bar{\theta}$, $\|\hat{V} - V^*\|_\infty < \epsilon$, i.e., the policy $\hat{\pi}$ is ϵ-optimal.

Proof: The way $\hat{\pi}$ and π^* are defined, we have $\hat{\pi}(i,\cdot) \to \pi^*(i,\cdot)$ as $\theta_0 \to \infty$. The claim now follows from Proposition 2 using the facts that $\max_i \|\bar{g}(i,\cdot)\|_\infty \leq B < \infty$ for some scalar $B > 0$, and that $\gamma \in [0,1)$. ■

B. Convergence of the Slower Recursion

The policy update in this algorithm is on the faster time scale and sees the value update as quasi-static. We have the following important result.

Lemma 4: Given $\epsilon > 0$, there exists $\bar{\theta}$ sufficiently large such that for $\theta_0 > \bar{\theta}$ and $V \approx V_n$ (i.e., V is tracking V_n on a slower time scale), recursion (7) on the slower timescale satisfies

$$\max_i \left| \min_u \sum_j p(i,u,j)(g(i,u,j) + \gamma V(j)) - V(i)) \right| < \epsilon,$$

for $n \geq 0$.

Proof: Follows as in Lemma 5.12 of [9]. ■

It follows that the iteration for V_n on the slower time scale satisfies

$$V_{n+1}(i) \approx V_n(i) + \bar{a}(n) \min_u [\textstyle\sum_j (p(i,u,j) g(i,u,j) + \gamma V_n(j)) - V_n(i) + N_{n+1}(i))],$$

where, for any $i \in S, n \geq 0$,

$$N_{n+1}(i) := g(i, \phi_n(i), \xi_n(i, \phi_n(i))) + \gamma V_n(\xi_n(i, \phi_n(i))) - E[g(i, \phi_n(i), \xi_n(i, \phi_n(i))) + \gamma V_n(\xi_n(i, \phi_n(i)))|\mathcal{F}_n],$$

is a martingale difference sequence, with the change from the previous definition of N_n being that we use $V_n(\cdot)$ here in place of $V(\cdot)$ The approximate equality '$\approx$' accounts for the error

$$E[g(i, \phi_n(i), \xi_n(i, \phi_n(i))) + \gamma V_n(\xi_n(i, \phi_n(i)))|\mathcal{F}_n] - \min_u [\sum_j (p(i,u,j) g(i,u,j) + \gamma V_n(j))],$$

which is seen from Lemma 4 to be $O(\epsilon)$.

We thus have a result similar to Theorem 5.13, [9]:

Theorem 5: Given $\epsilon > 0$, $\exists\ \bar{\theta} \equiv \bar{\theta}(\epsilon) > 0$ such that for all $\theta_0 \geq \bar{\theta}$, (V_n, θ_n), $n \geq 0$, governed according to (7)-(8), converges almost surely to the set

$$\{(V_{\pi_\theta}, \theta) \mid V_\gamma(i) \leq V_{\pi_\theta}(i) \leq V_\gamma(i) + \epsilon, \forall i \in S\},$$

where $V_\gamma(i), i \in S$ is the unique solution to the Bellman equation (4).

V. Numerical Results

We show here the results of experiments to study the empirical performance of the CA algorithm (7)-(8) and its comparison with the AC algorithm (Algorithm 3 of [9]). We compare these algorithms initially in tabular form so as to put both on a common footing and so that the comparison is legitimate, and also because we have shown the convergence analysis in tabular form. The difficulty in practice with the tabular form is its high dimensionality which can put it at a disadvantage, so the practical usage is invariably with function approximation. We also provide some comparisons with function approximations, but because of the possibility of multiple equilibria etc., there are other factors playing a role, so it is less conclusive evidence than the tabular form.

We present experiments on Grid World settings of various sizes and dimensions for different step-size schedules. For finite state-action MDP, there is always a unique optimal value function that we compute for our experiments. Our main performance metric is thus the value function error as obtained by the two algorithms, viz., the Euclidean distance between the running value function estimates from each algorithm and the optimal value function. We also study here performance comparisons using the running average cost as a function of the number of iterates. The figures show the performance comparisons in terms of (a) the value error estimates and (b) the running average cost obtained from the two algorithms. We also plot alongside in each figure the tail of the plots separately (after removing initial transients) for a better depiction of the performance of the two algorithms when closer to convergence. Finally, we compare the amount of computational time required by the two algorithms on each of the settings. All results are shown by averaging over 5 different runs of each algorithm with different initial seeds. The standard error from these runs is also shown.

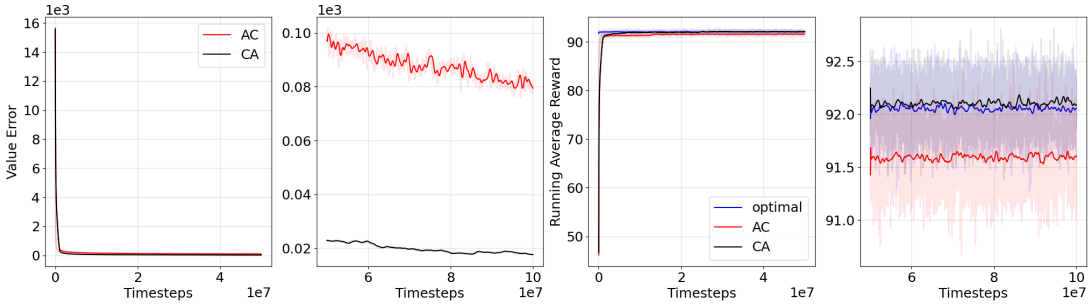

Fig. 1. Experiment 1: $|S| = 1000, |U| = 6, \alpha_1 = 1, \beta_1 = 0.55, \alpha_2 = 1, \beta_2 = 0.55$

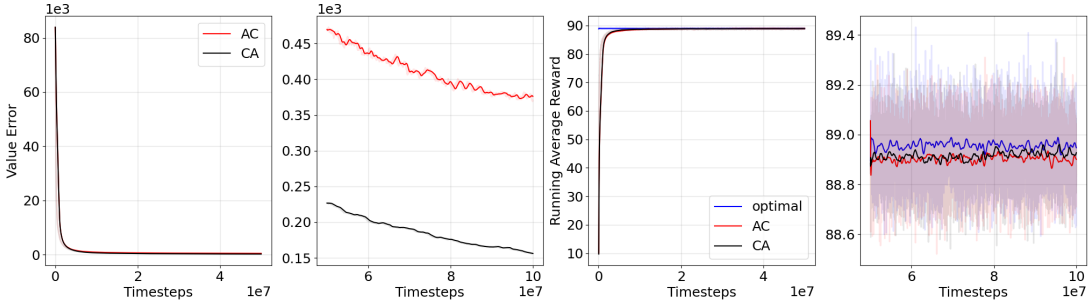

Fig. 2. Experiment 5: $|S| = 10000, |U| = 8, \alpha_1 = 0.75, \beta_1 = 0.55, \alpha_2 = 0.95, \beta_2 = 0.75$

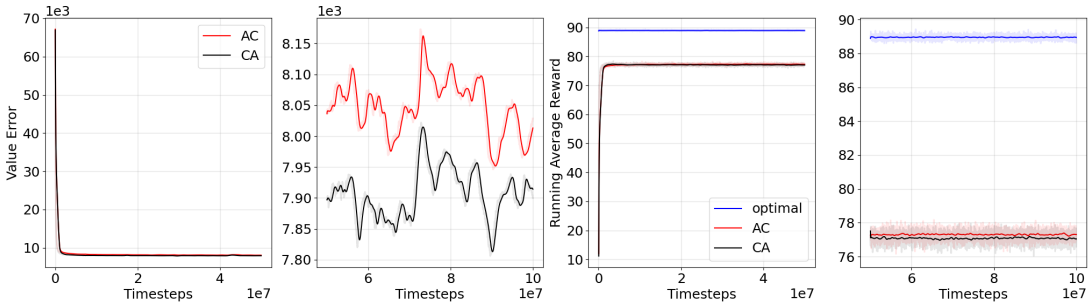

Fig. 3. Experiment 7: $|S| = 10000, |U| = 8, \alpha_1 = 0.95, \beta_1 = 0.75, \alpha_2 = 0.75, \beta_2 = 0.55$

Exp.	Value Error		Average Reward		Computational Time (hours)	
	AC	CA	AC	CA	AC	CA
1	79.33 ±4.23	17.57 ±1.59	91.59 ±1.13	92.21 ±0.41	11.18 ±0.36	11.23 ±0.26
2	41.27 ±4.63	41.49 ±4.00	91.67 ±0.90	92.18 ±0.49	11.25 ±0.22	11.28 ±0.32
3	80.88 ±5.15	21.88 ±0.85	91.73 ±0.70	91.72 ±0.49	10.87 ±0.20	11.28 ±0.30
4	24.78 ±1.64	23.38 ±2.22	86.53 ±0.34	86.01 ±1.28	11.05 ±0.20	10.97 ±0.06
5	370.12 ±5.08	155.51 ±4.04	88.71 ±0.18	88.84 ±0.25	20.80 ±0.30	20.98 ±0.81
6	2825.67 ±1.63	2824.96 ±0.72	75.96 ±0.42	76.11 ±0.71	28.30 ±0.43	28.91 ±0.93
7	8028.80 ±41.38	7899.68 ±51.49	77.28 ±0.43	76.94 ±0.46	53.17 ±1.12	52.39 ±0.50

TABLE I

MEAN AND STANDARD DEVIATION AFTER 10^8 TIME-STEPS OF AVERAGE REWARD, VALUE ERROR AND COMPUTATIONAL TIME

We conducted seven sets of experiments labelled Exp. 1 to 7. Experiments 1 to 5 are for the tabular setting as described in the paper. Experiments 6 and 7, on the other hand, are conducted with function approximation. Specifically, Experiment 6 is for the case when a linear function approximation architecture is used to approximate the value function, while Experiment 7 does so with a neural network architecture. All these experiments are conducted for various grid world settings. Specifically, Experiments 1 – 3 are on a 3-dimensional grid world of size $10 \times 10 \times 10$ (1000 states) and 6 actions (+x,-x,+y,-y,+z,-z). Experiment 4 is for the case of a 2-dimensional grid of size 20×20 and 4 actions (+x,-x,+y,-y). Experiment 5 is conducted on a 4-dimensional grid of size 10 in each dimension (10,000 states) and 8 actions. In the experiments for the tabular setting, we use step-sizes of the form $a(n) = \frac{1}{(\lfloor \frac{n}{100} \rfloor + 1)^{\alpha_1}}, b(n) = \frac{1}{(\lfloor \frac{n}{100} \rfloor + 1)^{\beta_1}}$ for AC and $a(n) = \frac{1}{(\lfloor \frac{n}{100} \rfloor + 1)^{\alpha_2}}, b(n) = \frac{1}{(\lfloor \frac{n}{100} \rfloor + 1)^{\beta_2}}$ for CA. We decrease the step-size every 100 time-steps in both algorithms for faster convergence. These step-sizes follow Assumptions 1(i)-(iii) but not Assumptions 1(iv)-(v) and 2(ii). We observe similar performance patterns when step-sizes that fully satisfy Assumptions 1 and 2 are used. The results of these experiments are shown in Figures 10-11 of [6] and discussed in detail there[2]. This indicates that CA is a promising new algorithm. For Experiment 6 (linear function approximation), we decrease the step-sizes every 100 time-steps (same as in the tabular case). We take features for state $i \in S$ so that the value at the $\lfloor \frac{i}{10} \rfloor$-th position is 1 and the rest are 0. Note that the basis functions are linearly independent (see [14]). In Experiment 7 (neural network architecture), we decrease the step-size at every time-step in both algorithms. This ensures good performance for both. We use, for this experiment, a fully connected feedforward neural network with two hidden layers, 10 neurons per layer, $\tanh$ activation function, and the single number state as input. We run all our experiments for 10^8 time-steps. The mean and standard deviation values from 5 runs after 10^8 time-steps for the various experiments are shown in Table I in the form $\mu \pm \sigma$, where μ and σ denote the mean and standard deviation respectively. For lack of space, we show here the plots for Experiments 1, 5 and 7, respectively, in Figures 1–3. Plots for the remaining experiments can be found in [6]. From Figures 1-2 and the other plots in [6] (see Figures 1–5 there), we see that the value function error successfully goes to almost zero. From the figures and the table, it is seen that the performance of our Critic-Actor algorithm is as good (in fact, slightly better overall) as the well-studied Actor-Critic algorithm. In [6], we also show the results of experiments where we compare the performance of the CA, AC and Q-learning algorithms when function approximation is used in each. We use policy gradient for the actor update and TD(0) for the critic update. The experiments are run on a two-dimensional grid of size 100×100, and 9 actions. Figures 8 and 9 in [6] show the plots for the cases when linear function approximation and neural network based approximators are used respectively for each algorithm. Both CA and Q-Learning show similar performance in Figure 8 and both algorithms show superior performance than AC. From Figure 9, all three algorithms show similar performance except that DQN takes significantly more computational time than CA and AC. We also see greater variance in the performance of AC.

[2]One cannot infer numerical superiority of one algorithm over another merely by the number of states and actions in the setting because whereas actor is the faster recursion in CA, critic is slower when compared to AC.

VI. Conclusions

We presented for the first time an important and previously unstudied class of algorithms – the critic-actor algorithms, that holds much promise. Like actor-critic, these are also two-timescale algorithms, however, where the value critic is run on a slower timescale as compared to the policy actor. Whereas actor-critic algorithms emulate policy iteration, we argue that critic-actor mimics value iteration. We proved the convergence of the algorithm. Further, we showed the results of several experiments on a range of Grid World settings and observed that our critic-actor algorithm shows similar or slightly better performance as compared to the corresponding actor-critic algorithm. We hope that our work will result in further research in this hitherto unexplored direction. In particular, it would be of interest to study further the critic-natural actor algorithms such as [5] with time scales reversed.

References

[1] Barto, A. G., Sutton, R. S. and Anderson, C. W., 1983. "Neuronlike adaptive elements that can solve difficult learning control problems". IEEE transactions on Systems, Man and Cybernetics, (5), pp. 834-846.

[2] Berstekas, D. P., 2019. *Reinforcement Learning and Optimal Control.* Athena Scientific.

[3] Bhatnagar, S. and Babu, K. M., 2008. "New algorithms of the Q-learning type". Automatica, 44(4), pp. 1111-1119.

[4] Bhatnagar, S. and Lakshmanan, K., 2016. "Multiscale Q-learning with linear function approximation". Discrete Event Dynamic Systems, 26(3), pp. 477-509.

[5] Bhatnagar, S., Sutton, R., Ghavamzadeh, M., and Lee, M., 2009. "Natural actor-critic algorithms". Automatica, 45(11), pp. 2471-2482.

[6] Bhatnagar, S., Borkar, V. S., and Guin, S. 2022. "Actor-Critic or Critic-Actor? A Tale of Two-Timescales". ArXiv Preprint arXiv preprint arXiv:2212.10477.

[7] Borkar, V. S., 1998. "Asynchronous stochastic approximation". SIAM Journal on Control and Optimization, 36(3), pp. 840-851. (Erratum in SIAM Journal on Control and Optimization, 38(2), 2000, pp. 662-663).

[8] Borkar, V. S., 2022. *Stochastic Approximation: A Dynamical Systems Viewpoint* (2nd edition), Hindustan Publishing Agency and Springer.

[9] Konda, V.R. and Borkar, V.S., 1999. "Actor-critic–type learning algorithms for Markov decision processes". SIAM Journal on control and Optimization, 38(1), pp. 94-123.

[10] Konda, V. R. and Tsitsiklis, J. N., 2003. "On actor-critic algorithms". SIAM journal on Control and Optimization, 42(4), pp. 1143-1166.

[11] Puterman, M. L., 2014. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley and Sons.

[12] Sutton, R., 1988. "Learning to predict by the method of temporal differences". Machine Learning, 3, pp. 9-44.

[13] Sutton, R., McAllester, D., Singh, S., and Mansour, Y., 1999. "Policy gradient methods for reinforcement learning with function approximation". Proceedings of NeurIPS, pp. 1057-1163.

[14] Tsitsiklis, J. N. and Van Roy, B., 1997. "An analysis of temporal difference learning with function approximation", IEEE Transactions on Automatic Control, 42(5), pp. 674-690.

[15] Watkins, C. J. C. H. and Dayan, P., 1992. "Q-learning". Machine Learning, 8, pp. 279-292.