A MATHEMATICAL ANALYSIS OF ACTOR-CRITIC ARCHITECTURES FOR LEARNING OPTIMAL CONTROLS THROUGH INCREMENTAL DYNAMIC PROGRAMMING*

Ronald J. Williams and Leemon C. Baird, III
College of Computer Science
Northeastern University
Boston, MA 02115

Abstract

Combining elements of the theory of dynamic programming with features appropriate for on-line learning has led to an approach Watkins has called *incremental dynamic programming*. Here we adopt this incremental dynamic programming point of view and obtain some preliminary mathematical results relevant to understanding the capabilities and limitations of *actor-critic* learning systems. Examples of such systems are Samuel's learning checker player, Holland's bucket brigade algorithm, Witten's adaptive controller, and the adaptive heuristic critic algorithm of Barto, Sutton, and Anderson. Particular emphasis here is on the effect of complete asynchrony in the updating of the actor and the critic across individual states or state-action pairs. The main results are that, while convergence to optimal performance is not guaranteed in general, there are a number of situations in which such convergence is assured.

1 Introduction

The purpose of this paper is to present some results in the theory of *incremental dynamic programming* (Watkins, 1989), which combines elements of the theory of dynamic programming with features appropriate for on-line learning in the absence of an *a priori* model of the environment. An excellent discussion of this approach is provided by Barto, Sutton, and Watkins (1989), who point out that it may be regarded as a direct method of adaptive control. Another investigator who has emphasized the potential role of dynamic-programming-based strategies in biological and artificial learning systems is Werbos (1987), who has used the term *heuristic dynamic programming* for his approach.

The results presented here concern the use of *actor-critic* learning systems, which consist of two jointly adaptive modules. One module, called the *actor*, implements the current policy but also learns to improve its choice of action in order to optimize the reward provided by the other module, which is called the *critic*. The reward provided by the critic is intended to represent an estimate of the expected long-term external reward, but the critic must also adapt in order both to improve the accuracy of its estimate given the current policy and to keep up with the ever-changing policy as the actor learns. There are a number of related algorithms that can be considered to have this particular form, including that used in Samuel's (1957) learning checker player, Holland's (1986) bucket brigade algorithm, Witten's (1977) adaptive controller, and the adaptive heuristic critic algorithm of Barto, Sutton, and Anderson (1983). Sutton (1984; 1988) appears to be one of the first to call attention to a common basis for these algorithms.

The main question addressed by the results presented here concerns the effect of asynchronous adaptation of the actor and critic modules at individual states (or state-action pairs, as appropriate). That is, we ask: What happens if the actor and critic are updated state by state in no particular order? Will such joint incremental adaptation always succeed at finding optimal policies, or can it sometimes fail? Ignored here are a number of important additional issues we intend to address in later work, including problems of dealing with unknown stochastic environments. Here we simply assume that all transition probabilities and expected rewards can be determined exactly from interaction with the initially unknown environment. This includes as a special case the situation when all transitions and rewards are deterministic, so the reader may prefer to imagine that this is

*We gratefully acknowledge the substantial contributions to this effort provided by Andy Barto, who sparked our original interest in these questions and whose continued encouragement and insightful comments and criticisms have helped us greatly. Special thanks also to Rich Sutton, who has influenced our thinking on this subject in numerous ways. This work was partially supported by Grant IRI-8921275 from the National Science Foundation.

the case we are actually treating.[1] Another simplifying assumption we make is to consider arbitrary sequences of application of certain one-state-at-a-time operators to actor-critic systems without regard for how those states come to be visited by the learning system.

2 Mathematical Preliminaries

Here we give the mathematical formulation of the type of problem addressed in this paper and introduce appropriate definitions and notational conventions. We also summarize some standard computational strategies from the theory of dynamic programming and describe additional elementary results relevant to the incremental point of view adopted here.

2.1 Markov Environment

Let X be a finite state space and A a finite action space. We let $\Pi = \{\pi : X \to A\}$ denote the set of stationary, non-randomized policies, with $V = \{v : X \to \mathcal{R}\}$ denoting the set of value functions. For any finite set S we denote its cardinality by $|S|$.

We assume given a Markov environment, and we let $f(x,a)$ denote the randomly determined successor of state x when action a is chosen. The behavior of this random next-state function is determined by the transition probabilities $p^a_{xy} = Pr\{f(x,a) = y\}$ for $x, y \in X$ and $a \in A$. We also assume that associated with each choice of action a at each state x is a randomly determined immediate reward $r(x,a)$, with $R(x,a) = E\{r(x,a)\}$ denoting its expected value.

2.2 Total Expected Discounted Return

The results presented here are all based on the use of some fixed discounting parameter $\gamma \in (0,1)$, which is used to define a measure of performance on policies as follows. For any policy π, define $v^\pi \in V$, the *total expected discounted return* (or just the *return*) for π, to be that value function assigning to state x the quantity

$$v^\pi(x) = E\left\{\sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)) \mid x_0 = x\right\}$$

where $x_0, x_1, x_2, \ldots$ is a random sequence of states determined by $x_0 = x$ and $x_{t+1} = f(x_t, \pi(x_t))$ for each $t \geq 0$.

We let v^* denote the *ideal evaluation function*, or the return from any optimal policy.

[1] Nevertheless, with the intent of eventually extending these results, we adhere throughout to the probabilistic formulation used in the theory of *stochastic dynamic programming* (Ross, 1983).

2.3 Markov Decision Task

For purposes of this paper we define a *(stationary, finite) Markov decision task*, to be a 5-tuple (X, A, r, f, γ) consisting of a finite state set X, finite action set A, stationary random reward function r, stationary random transition function f, and discount parameter γ. As discussed earlier, we adopt the assumption throughout that all transition probabilities and expected rewards can be determined exactly from interaction with the environment. This includes as a special case the situation when the transition function f and reward function r are deterministic.

2.4 Actor-Critic Architecture

We regard the internal state of an actor-critic architecture as simply an ordered pair $(\pi, v) \in \Pi \times V$. Whatever learning algorithm is used to adapt the actor is considered to modify the policy π, while learning in the critic alters the value function v. With the system initialized to some internal state (π_0, v_0), our interest is in analyzing properties of the succession of internal states that result from applying some particular learning algorithm (i.e., sequence of operators). In particular, we are concerned with the question of whether the system will eventually learn to perform optimally.

2.5 Local Backup Operators

For given $v \in V$, $x \in X$, and $a \in A$, define

$$\begin{aligned} L^v(x,a) &= E\{r(x,a) + \gamma v(f(x,a))\} \\ &= R(x,a) + \gamma \sum_{y \in X} p^a_{xy} v(y). \end{aligned}$$

This quantity represents a *one-step lookahead* value, or the expected reward that would be received if, starting in state x, action a is selected, giving rise to an immediate reward of $r(x,a)$, and then the process terminates with an additional reward of $\gamma v(f(x,a))$.

Then, for each $x \in X$ and $\pi \in \Pi$, define the operator $B^\pi_x : V \to V$ by

$$B^\pi_x v(y) = \begin{cases} L^v(x, \pi(x)) & \text{if } y = x \\ v(y) & \text{otherwise.} \end{cases}$$

That is, $B^\pi_x v$ is the value function obtained from v by replacing $v(x)$ by the one-step lookahead value along π. Also, for each $x \in X$, define the operator $B_x : \Pi \times V \to \Pi \times V$ by

$$B_x(\pi, v) = (\pi, B^\pi_x v).$$

We refer to operators of either form as *local backup operators*.

2.6 Local Policy Improvement Operators

For each $x \in X$, $a \in A$, and $v \in V$, define the operator $I^v_{x,a} : \Pi \to \Pi$ by

$$I^v_{x,a}\pi(y) = \begin{cases} a & \text{if } y = x \\ & \text{and } L^v(x,a) \geq L^v(x,\pi(x)) \\ \pi(y) & \text{otherwise.} \end{cases}$$

Also, for each $x \in X$ and $a \in A$, define the operator $I_{x,a} : \Pi \times V \to \Pi \times V$ by

$$I_{x,a}(\pi, v) = (I^v_{x,a}\pi, v).$$

In addition, for each $x \in X$ and $v \in V$, define the operator $I^v_x : \Pi \to \Pi$ by

$$I^v_x\pi(y) = \begin{cases} a & \text{if } y = x \\ & \text{and } L^v(x,a) = \max_{\alpha \in A} L^v(x,\alpha) \\ \pi(y) & \text{otherwise,} \end{cases}$$

where it is understood that some method is used to choose among several equal candidates for giving the maximum one-step lookahead value. Also, for each $x \in X$, define the operator $I_x : \Pi \times V \to \Pi \times V$ by

$$I_x(\pi, v) = (I^v_x\pi, v).$$

We refer to any of these operators as *local policy improvement operators*. The operator $I_{x,a}$ alters the policy at state x by comparing the lookahead values for the current policy and for action a while I_x considers all possible actions. As we discuss below, I_x can be viewed as representing a step in both the value iteration and policy iteration procedures. The reason we also wish to consider the $I_{x,a}$ operators is that they represent an even finer-grained incrementality, thus conforming to the overall spirit of our approach. In some applications, particularly when there are a large number of actions, it may be difficult or computationally infeasible to compare all possible actions. Related to this is the possibility that the need for fast reaction may preclude consideration of all but some small number of alternative actions at any one time.

Note that both the $I_{x,a}$ and I_x operators make changes to a policy based on comparison of the lookahead values of two or more actions. In some situations it may not be realistic to assume that even this is possible. Later we will introduce additional operators which do not require comparing two or more lookahead values in this way but instead involve comparing a lookahead value with the current value for a state.

2.7 Convergence to Optimality Under Asynchronous Operator Application

The results presented in this paper all deal with problems having the following general form. We assume given a set $\mathcal{S}$ of operators mapping $\Pi \times V \to \Pi \times V$, and we apply a sequence $T_1, T_2, \ldots$ of these operators in succession, beginning with an initial policy-value pair $(\pi_0, v_0) \in \Pi \times V$. This gives rise to a sequence of policy-value pairs $(\pi_0, v_0), (\pi_1, v_1), (\pi_2, v_2), \ldots$, where $(\pi_k, v_k) = T_k(\pi_{k-1}, v_{k-1})$ for all $k > 0$. For any such set $\mathcal{S}$, we say that a result holds under *asynchronous application of (the operators from)* $\mathcal{S}$ if it holds for any sequence of operators from $\mathcal{S}$ in which each operator appears infinitely often. In particular, we want to identify situations when the limiting behavior of the system is optimal, which we formulate as follows: Let us say that such a sequence of policy-value pairs *converges to optimality* if $\lim_{i\to\infty} v_i = v^*$ and there exists M such that every π_i is optimal when $i \geq M$.

In what follows we shorten the notation for the various sets of operators under consideration by always regarding any set of subscripted operators to mean the set of all such operators. For example, $\{B_x I_{x,a}\}$ is short for $\{B_x I_{x,a} \mid x \in X, a \in A\}$.

2.8 Computation of the Return for a Stationary Policy

Since $v^\pi(x) = L^{v^\pi}(x, \pi(x))$ for all states x when π is a stationary policy, v^π can be computed by solving the system of $|X|$ linear equations

$$v^\pi(x) = R(x, \pi(x)) + \gamma \sum_{y \in X} p^{\pi(x)}_{xy} v^\pi(y).$$

This system can be solved in a number of ways, but most interesting from the point of view of incremental dynamic programming are techniques that begin with an arbitrary value function v and involve iterative updating of the values $\{v(x) \mid x \in X\}$ by replacing each by $L^{v^\pi}(x, \pi(x))$. The following result, whose easy proof we omit, shows that the correct result is obtained even if this updating is performed completely asynchronously across states.

Proposition 1 *If $\{B_x \mid x \in X\}$ is applied asynchronously to any arbitrary (π_0, v_0), then $v_i \to v^{\pi_0}$ as $i \to \infty$.*

2.9 Policy Iteration

The *policy iteration* procedure determines an optimal policy by starting with any initial policy π_0 and repeating the following step: For policy π_k, compute its return v^{π_k} and then define policy π_{k+1} to satisfy

$$L^{v^{\pi_k}}(x, \pi_{k+1}(x)) = \max_\alpha L^{v^{\pi_k}}(x, \alpha)$$

for each x. The process terminates as soon as $v_k(x) = v_{k+1}(x)$ for all x.

We can use Proposition 1 to define a sequence of operators to be applied to an initial (π_0, v_0) that essentially carries out the policy iteration strategy, once we observe that it is only necesssary to compute v^π to a sufficiently close approximation in the policy iteration procedure since X and A are finite. It then follows that application of this particular sequence of operators to (π_0, v_0) gives rise to a sequence of policy-value pairs that converges to optimality.

2.10 Value Iteration

The *value iteration* procedure determines an optimal policy by first computing v^*. Any policy π for which

$$L^{v^*}(x, \pi(x)) = \max_\alpha L^{v^*}(x, \alpha)$$

will then be an optimal policy. That is, an optimal policy is determined by choosing for each x the action giving the maximum one-step lookahead value when v^* is used to determine the evaluation of the successor states for x.

The ideal evaluation function v^* is the unique solution to the system of $|X|$ nonlinear equations

$$v^*(x) = \max_{\alpha \in A} \left\{ R(x, \alpha) + \gamma \sum_{y \in X} p^\alpha_{xy} v^*(y) \right\}.$$

The value iteration procedure for solving these equations involves starting with an arbitrary value function v and then iteratively updating the values $\{v(x) \mid x \in X\}$ by replacing each by $\max_\alpha L^v(x, \alpha)$. Watkins (1989) has proved that this value iteration method may be carried out completely asynchronously across states and has used it as the basis for a particular algorithm called *Q-learning*. We omit the easy proof of this result and restate it in the following form, which is more appropriate for our purposes.

Proposition 2 *Asynchronous application of the set of operators $\{B_x I_x\}$ to an arbitrary (π_0, v_0) yields convergence to optimality.*

3 Main Results

The various ways of carrying out asynchronous value iteration and the method described above for carrying out policy iteration amount to application of certain constrained sequences of local backup and policy improvement operators, any of which give rise to convergence to optimality. These particular sequences can be interpreted as enforcing certain forms of coordination when adapting the components of an actor-critic system. Here we consider what happens if such coordination is not enforced. In the interest of brevity, we give all results without proof.

3.1 Use of Policy Operators That Compare Multiple Actions

The following is a key result.

Theorem 1 *Suppose that an arbitrary sequence of operators from $\{B_x\} \cup \{I_{x,a}\}$ in which each operator appears infinitely often is applied to some (π_0, v_0). Then, if $\lim_{i \to \infty} v_i(x)$ exists for each $x \in X$, the resulting sequence of policy-value pairs converges to optimality.*

We also have the following collection of negative results.

Theorem 2 *Convergence to optimality need not occur under asynchronous application of any of the following sets of operators:*
(a) $\{B_x\} \cup \{I_x\}$;
(b) $\{B_x\} \cup \{B_x I_x\}$;
(c) $\{B_x I_{x,a}\}$;
(d) $\{I_x B_x\}$;
(e) $\{I_{x,a} B_x\}$.

By Theorem 1, the only way such convergence to optimality can fail to occur is when the sequence of value functions approaches no limit. Theorem 2 is proved by exhibiting particular Markov decision tasks, initial policy-value pairs, and appropriate sequences of operators for which the resulting sequence of policy-value pairs cycles endlessly. In fact, there are two Markov environments, each having 6 states and 2 actions, which can be used for all 5 cases.

Theorem 1 is useful for identifying sequences of local backup and policy improvement operators and initial policy-value pairs such that convergence to optimality is guaranteed, as in the following result.

Theorem 3 *If $L^{v_0}(x, \pi_0(x)) \geq v_0(x)$ for all $x \in X$, then asynchronous application of $\{B_x\} \cup \{I_{x,a}\}$ to (π_0, v_0) yields convergence to optimality.*

This result follows from Theorem 1 since it can be shown that the sequence $v_0(x), v_1(x), v_2(x), \ldots$ is nondecreasing and bounded above for each x. We single out the following two noteworthy corollaries to this theorem, the second of which provides an interesting generalization of policy iteration.

Corollary 1 *If all the expected rewards $R(x, a)$ are nonnegative and $v_0(x) = 0$ for all x, then asynchronous application of $\{B_x\} \cup \{I_{x,a}\}$ to (π_0, v_0) yields convergence to optimality.*

Corollary 2 *If $v_0 = v^{\pi_0}$, then asynchronous application of $\{B_x\} \cup \{I_{x,a}\}$ to (π_0, v_0) yields convergence to optimality.*

Another way to guarantee convergence of the value function is to impose certain conditions on the discount parameter γ, which leads to other interesting ways that convergence to optimality may be assured.

Theorem 4 *Asynchronous application of the set $\mathcal{S}$ of operators to an arbitrary (π_0, v_0) yields convergence to optimality if:*
(a) $\mathcal{S} = \{B_x\} \cup \{I_x\}$ and $\gamma < 1/2$; or
(b) $\mathcal{S} = \{B_x\} \cup \{I_{x,a}\}$ and $\gamma < 1/|A|$.

Since it is common to use a discount parameter having a value reasonably close to 1, this result would not appear to be particularly useful. Nevertheless, we will see that it has interesting consequences for more practical values of γ when we consider the use of multi-step backup operators below.

The following result shows that counterexamples like those used to establish Theorem 2 are rare, in some sense.

Theorem 5 *Suppose that a sequence of operators is obtained by independent random sampling from a fixed distribution over $\{B_x\} \cup \{I_{x,a}\}$ such that every operator has nonzero probability. Then application of this sequence of operators to any initial policy-value pair (π_0, v_0) yields convergence to optimality with probability 1.*

3.2 Use of Policy Operators That Examine Single Actions

Now we consider alternative policy improvement operators that do not require comparing two or more lookahead values in order to make changes to a policy. These operators work by comparing the lookahead value along some candidate action with the current value for the state in question.

For each $x \in X$, $a \in A$, and $v \in V$, define the operator $J^v_{x,a} : \Pi \to \Pi$ by

$$J^v_{x,a}\pi(y) = \begin{cases} a & \text{if } y = x \text{ and } L^v(x,a) \geq v(x) \\ \pi(y) & \text{otherwise.} \end{cases}$$

Also, for each $x \in X$ and $a \in A$, define the operator $J_{x,a} : \Pi \times V \to \Pi \times V$ by

$$J_{x,a}(\pi, v) = (J^v_{x,a}\pi, v).$$

Theorem 6 *If $L^{v_0}(x, \pi_0(x)) \geq v_0(x)$ for all $x \in X$, then asynchronous application of $\{B_x\} \cup \{J_{x,a}\}$ to (π_0, v_0) yields convergence to optimality.*

We can construct an algorithm roughly analogous to asynchronous value iteration but requiring no comparisons of multiple actions by combining the use of the J operators with the use of certain conditional backup operators, which we now define. We first define $C_{x,a} : V \to V$ by

$$C_{x,a}v(y) = \begin{cases} \max\{v(x), L^v(x,a)\} & \text{if } y = x \\ v(y) & \text{otherwise,} \end{cases}$$

and then, in a harmless abuse of notation, we further define $C_{x,a} : \Pi \times V \to \Pi \times V$ by

$$C_{x,a}(\pi, v) = (C_{x,a}\pi, v).$$

Theorem 7 *If $v_0(x) < v^*(x)$ for all states x, then asynchronous application of $\{C_{x,a}J_{x,a}\}$ to (π_0, v_0) yields convergence to optimality.*

3.3 Use of Multi-Step Backup Operators

It is convenient in what follows to abuse notation slightly and define $L^v(x, \pi)$, where π is a policy, to mean the same as $L^v(x, \pi(x))$. We then extend this notion as follows. For given $v \in V$, $x \in X$, $\pi \in \Pi$, and positive integer n, define

$$L^{v,n}(x, \pi) = E\left\{\sum_{t=0}^{n-1} \gamma^t r(x_t, \pi(x_t)) + \gamma^n v(x_n)\right\}$$

where $x_0, x_1, x_2, \ldots, x_n$ is a random finite sequence of states determined by $x_0 = x$ and $x_{t+1} = f(x_t, \pi(x_t))$ for each $t \in [0, n)$. This is an n-step generalization of the one-step lookahead value $L^{v,1}(x, \pi) = L^v(x, \pi)$. Note also that it represents an approximation to $v^\pi(x)$ in which the tail of the infinite series is replaced by $\gamma^n v(x_n)$.

Then, for each $x \in X$, $\pi \in \Pi$, and positive integer n, define the operator $B^{\pi,n}_x : V \to V$ by

$$B^{\pi,n}_x v(y) = \begin{cases} L^{v,n}(x, \pi) & \text{if } y = x. \\ v(y) & \text{otherwise.} \end{cases}$$

That is, $B^{\pi,n}_x v$ is the value function obtained from v by replacing $v(x)$ by the n-step lookahead value along π. Also, for each $x \in X$, define the operator $B^{(n)}_x : \Pi \times V \to \Pi \times V$ by

$$B^{(n)}_x(\pi, v) = (B^{\pi,n}_x \pi, v).$$

We call operators of either form *local multi-step backup operators.*

The next result is a generalization of Theorem 4.

Theorem 8 *Asynchronous application of the set $\mathcal{S}$ of operators to an arbitrary (π_0, v_0) yields convergence to optimality if:*
(a) $\mathcal{S} = \left\{B^{(n)}_x\right\} \cup \{I_x\}$ and $\gamma^n < 1/2$; or
(b) $\mathcal{S} = \left\{B^{(n)}_x\right\} \cup \{I_{x,a}\}$ and $\gamma^n < 1/|A|$.

4 Summary

We have presented results here that address questions of convergence to optimality when the actor and critic are updated completely asynchronously across individual states (or, in some cases, state-action pairs). The main results are that, while there are cases when such convergence may fail, there are a number of reasonably general situations in which such convergence is guaranteed. While these results should be regarded as preliminary because of the simplifying assumptions made in their derivation, it is interesting to note that there is at least one interesting class of algorithms to which these results are directly applicable in their current form—namely, certain forms of Sutton's (1990) *relaxation planning* algorithms, in which learning occurs while arbitrarily selected state-action pairs are explored in a model that is gradually acquired on line. Furthermore, these results provide insights into the potential range of asynchronous actor-critic adaptation behaviors, and we believe that this set of preliminary results can serve as the basis for deriving further results applicable to other useful learning algorithms.

References

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics, 13*, 835-846.

Barto, A. G., Sutton, R. S., & Watkins, C. J. C. H. (1989). *Learning and sequential decision making* (COINS Technical Report 89-95). Amherst, MA: University of Massachusetts, Department of Computer and Information Science.

Holland, J. H. (1986). Escaping brittleness: The possibility of general-purpose learning algorithms applied to rule-based systems. In: R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.) *Machine Learning: An Artificial Intelligence Approach, Volume II.* Los Altos, CA: Morgan Kaufmann.

Ross, S. 1983). *Introduction to Stochastic Dynamic Programming.* New York: Academic Press.

Samuel, A. L. (1957). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development, 3*, 210-229. Reprinted in: E. A. Feigenbaum and J. Feldman (Eds.) (1963). *Computers and Thought.* New York: McGraw-Hill.

Sutton, R. S. (1984). *Temporal credit assignment in reinforcement learning.* Ph.D. Dissertation, University of Massachusetts, Amherst (also COINS Technical Report 84-02).

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

Sutton, R. S. (1990). First results with DYNA, an integrated architecture for learning, planning, and reacting. *Stanford Spring Symposium on Planning.*

Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* Ph.D. Dissertation, Cambridge University, Cambridge, England.

Werbos, P. J. (1987). Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics, 17*, 7-20.

Witten, I. H. (1977). An adaptive optimal controller for discrete-time Markov environments. *Information and Control, 34*, 286-295.