Pattern Recognizing
Stochastic Learning Automata

Andrew. G. Barto and P. Anandan

COINS Technical Report 84–30
December 1984

Abstract

A class of learning tasks is described that combines aspects of learning automaton tasks and supervised learning pattern-classification tasks. We call these tasks ***associative reinforcement learning tasks.*** An algorithm is presented, called the ***associative reward-penalty***, or A_{R-P} , ***algorithm***, for which a form of optimal performance is proved. This algorithm simultaneously generalizes a class of stochastic learning automata and a class of supervised learning pattern-classification methods related to the Robbins-Monro stochastic approximation procedure. The relevance of this hybrid algorithm is discussed with respect to the collective behavior of learning automata and the behavior of networks of pattern-classifying adaptive elements. Simulation results are presented that illustrate the associative reinforcment learning task and the performance of the the A_{R-P} algorithm as compared with that of several existing algorithms.

This research was supported by the Air Force Office of Scientific Research and the Avionics Laboratory (Air Force Wright Aeronautical Laboratories) through contract F33615-83-C-1078. The authors wish to thank R. Sutton and C. Anderson for contributing in many ways to the development of results reported in this article. Most particularly, the content of Appendix I is largely due to R. Sutton. They also wish to thank J. Horowitz and D. Geman for valuable consultation on the mathematics involved. Of course, any errors and omissions are our own.

I. INTRODUCTION

Among the many classes of tasks that have been formalized by theorists interested in learning are 1) the optimization problems under uncertainty focussed upon by those studying learning automata, and 2) the supervised learning pattern-classification tasks which have been studied extensively since the early 1960s. In this article we describe a class of learning tasks that combines these two standard types of problems, and we present an algorithm for which we prove a form of optimal performance in these hybrid tasks by extending Lakshmivarahan's [1], [2], proof for time-varying stochastic automata. We call this algorithm the *associative reward-penalty*, or A_{R-P}, algorithm. Under one set of restrictions, it specializes to a stochastic learning automaton algorithm, in particular, to a nonlinear reward-penalty algorithm of the non-absorbing type as characterized by Lakshmivarahan [2]. Under another set of restrictions, it specializes to the well-known perceptron algorithm of Rosenblatt [3] and, with a minor modification, to a supervised learning pattern-classification method based on the Robbins-Monro stochastic approximation procedure [4]. Our interest, however, is in the case in which both of these aspects of the algorithm operate simultaneously—yielding a pattern-classifying stochastic learning automaton.

This algorithm is the product of an effort to provide a formal basis for earlier results obtained by computer simulation [5]–[9]. Our interest in this algorithm is a result our simulation experiments with networks of adaptive components implementing similar algorithms, and we discuss the implications of this approach below. Such adaptive elements constitute "self-interested" network components, the study of which was suggested to us by the theory of memory, learning, and intelligence developed by Klopf [10], [11], in which is proposed a class of algorithms similar to that studied here. To the best of our knowledge, the closest approximation to the particular algorithm presented here is the "selective bootstrap adaptation" algorithm of Widrow, Gupta, and Maitra [12], and we discuss this algorithm below. Also related is the much earlier stochastic pattern-classifying adaptive element of Farley and Clark [13]. As we discuss below (and as extensively discussed in refs. [5] and [8]), later pattern-classification algorithms were designed for tasks that differ in an essential way from the task considered here. The A_{R-P} algorithm also has parallels with aspects of the "stimulus sampling theory" of mathematical psychology [14]–[17], although that research did not make contact with pattern-classification techniques. One study that combines stochastic automaton and pattern-classification algorithms is that of Jarvis [18], but the resulting algorithm is different from our own.

Before we define the learning tasks in which we are interested, which we call *associative reinforcement learning tasks*,[1] we briefly describe the most common learning automaton task, a commonly studied supervised learning pattern-classification task, and certain types of algorithms that have been applied to each.

II. LEARNING AUTOMATA

The theory of learning automata originated with the work of Tsetlin [19] and the independent research of mathematical psychologists [14]–[17], and has been extensively developed since then. Good reviews are provided by refs. [20] and [21]. Also relevant is the independently developed line of research concerning the "two-armed bandit problem" (e.g., refs. [22] and [23]). The framework in which learning automata have been most widely studied consists of an automaton and an environment connected in a feedback loop. The environment receives each action emitted by the automaton and produces a signal, which acts as input to the automaton, that evaluates the suitability of that action. The evaluative feedback, which usually indicates *success* or *failure*, is probabalistic—there can be a nonzero probability of either evaluation from the environment in response to any action of the automaton. It is assumed that nothing is known *a priori* about the probabilities with which the environment determines success and failure feedback. An automaton that tends to increase its expectation of success is said to be a learning automaton. Ideally, one wishes the automaton eventually to emit only the action corresponding to the largest probability of success.

More formally, at each time step k, the automaton selects an action, a_k, from a finite action set, $A = \{a^{(1)}, \ldots, a^{(r)}\}$. The environment receives a_k as input and sends to the

[1] In earlier publications by Barto and colleagues [5]–[9], these tasks were called ***associative search tasks***, but we use the present terminology to avoid confusion with the unrelated usage of the same term in the field of information retrieval.

automaton an evaluation $b_k \in B = \{1, -1\}$, where 1 and -1 respectively indicate *success* and *failure*. The environment is characterised by its input set A, output set B, and a set of success probabilities, $\{d^{(1)}, \ldots, d^{(r)}\}$, where $d^{(i)} = Pr\{b_k = 1 \mid a_k = a^{(i)}\}$. If these success probabilities remain constant over time, the environment is said to be *stationary*; otherwise, it is said to be *nonstationary*. The success probabilities are unknown from the start.

The class of learning automata relevant to this article consists of variable-structure stochastic (VSS) learning automata, which can be described as methods for updating action probabilities. At each time step k, the automaton selects an action from A according to a probability vector $p_k = (p_k^{(1)}, \ldots, p_k^{(r)})^{\mathrm{T}}$, where $p_k^{(i)} = Pr\{a_k = a^{(i)}\}$ and the $^{\mathrm{T}}$ indicates transpose. These automata implement a common-sense notion of reinforcement learning: If action $a^{(i)}$ is chosen and the environment's feedback indicates success, then $p^{(i)}$ is increased and the probabilities of the other actions are decreased; whereas if the feedback indicates failure, then $p^{(i)}$ is decreased and the probabilities of the other actions are increased. Many methods that have been studied are similar to the following *linear reward-penalty* (L_{R-P}) method:

If $a_k = a^{(i)}$ and the resultant evaluation is success (i.e., $b_k = 1$), then

$$\begin{aligned} p_{k+1}^{(i)} &= p_k^{(i)} + \alpha(1 - p_k^{(i)}) \\ p_{k+1}^{(j)} &= (1 - \alpha)p_k^{(j)}, \quad j \neq i. \end{aligned} \tag{1}$$

If $a_k = a^{(i)}$ and $b_k = -1$ (failure), then

$$\begin{aligned} p_{k+1}^{(i)} &= (1 - \beta)p_k^{(i)} \\ p_{k+1}^{(j)} &= \frac{\beta}{r - 1} + (1 - \beta)p_k^{(j)}, \quad j \neq i, \end{aligned} \tag{2}$$

where $0 < \alpha, \beta \leq 1$. When $\alpha = \beta$, the algorithm is the symmetric L_{R-P} algorithm, and when $\beta = 0$, it is called the *linear reward-inaction* (L_{R-I}) algorithm.

The behavior of learning automata is studied with respect to several measures of asymptotic performance [1], [20], [21]. The expected probability of success at step k

is

$$M_k = \sum_{i=1}^{r} d^{(i)} p_k^{(i)}. \tag{3}$$

Letting $M^\circ = \frac{1}{r}\sum_{i=1}^{r} d^{(i)}$ and $d^l = \max_i\{d^{(i)}\}$, a learning automaton is *expedient* if $\lim_{k\to\infty} E\{M_k\} > M^\circ$; *optimal* if $lim_{k\to\infty} E\{M_k\} = d^l$; ϵ*-optimal* if for every $\epsilon > 0$, algorithm parameters exist such that $\lim_{k\to\infty} E\{M_k\} \geq d^l - \epsilon$; and *absolutely expedient* if $E\{M_{k+1} - M_k \mid p_k\} > 0$ for all k and all $p_k^{(i)} \in (0,1)$.

A considerable body of theory exists for VSS automata in stationary environments. It is known, for example, that any L_{R-P} algorithm is expedient for any stationary environment and that the L_{R-I} algorithm is ϵ-optimal. Necessary and sufficient conditions for the ϵ-optimality of a large class of algorithms have been developed, and many algorithms satisfy these conditions. However, no algorithm has been correctly shown to be optimal. We refer the reader to refs. [20] and [21] for reviews of existing results.

For our purposes in this article, it is important to note that the only input received by a learning automaton from its environment is the success/failure signal. While this restriction serves well to focus attention on a pure form of decision under uncertainty, most practical problems permit the possiblity of using information other than the evaluation signal for improving performance. For example, in a nonstationary environment, a learning automaton of the type described above will continuously adjust its action probabilities as it tracks the time-varying optimal action. This is seen most clearly in "switched" and periodic environments in which the success probabilities are piecewise constant or periodic functions of time [19]. When the environmental probabilities change, a learning automaton's performance often decreases as it selects actions according to a probability vector that was formed under the influence of environmental success probabilities that are no longer present. On the other hand, if an automaton receives some indication of the environment's state, it can use this information to access action probabilities appropriate for acting when the environment is in that state.

The usual way to construct a learning automaton that can discriminate between environmental states is to have it maintain a separate action probability vector for each environmental state, and then use one of the algorithms described above to update the probability vector corresponding to the current state of the environment. This approach amounts to the use of a "lookup-table" for acquiring and accessing appropriate action probabilities. The table has an entry for each environmental state that can be discriminated. This approach has been used in problems with periodic environments [21]; in the control of Markov processes [1], [24]; and in the control of other types of systems, e.g., ref. [25]. Although it permits immediate extension of existing results about learning automata to allow dependency on environmental state, this lookup-table approach shares the problems inherent in any lookup-table method of storing a mapping: It can be costly in terms of memory since each state requires a separate probability vector, and, more importantly, it offers no possibility for generalizing among states in order to reduce storage requirements and increase learning rate. The algorithm we present here is based instead on parameterizing the action probability vector and constructing a mapping from vectors of input features representing environmental states to this parameter. In this regard, it is related to certain pattern-classification algorithms which we discuss next.

III. SUPERVISED LEARNING PATTERN CLASSIFICATION

In supervised learning pattern classification, as we shall use the term, the learning system's environment presents it with a sequence of vectors, thought of as patterns, together with a class label for each vector that indicates how that vector should be classified. A sequence of patterns paired with the class labels is a "training sequence," and the environment is often called a "teacher" or "supervisor." The learning system adjusts its decision rule in order to match the given class label for each member of the training sequence, or more generally, to minimize the probability of misclassification. Since the resulting decision rule also applies to patterns not in the training sequence, the system performs a type of generalization. It is this capacity for generalization that makes these methods useful for storing information more compactly than would be possible with table-lookup methods (*if* the form of generalization is appropriate for the problam at hand).

Supervised learning pattern classification has been studied intensely, and a good overview is provided in ref. [26]. Hinton and Anderson [27] discuss the use of these and related methods for associative information storage. Artificial Intelligence researchers study versions of this same type of problem as "learning from examples," "concept formation," or "inductive inference" [26]. Here we are interested in pattern-classification algorithms based on stochastic approximation methods. We follow the presentation given in refs. [4] and [26] and consider only the case of two pattern classes.

Suppose that the pattern-classification system's environment determines a training sequence by first selecting at each step a class ω_i, $i = 1, 2$, according to the *a priori* probability $Pr\{\omega_i\}$, and then selecting a pattern vector according to the class-conditional probability $Pr\{x|\omega_i\}$. For each x let z be its class label, with $z = 1$ if class ω_1 was selected and $z = -1$ if class ω_2 was selected. The training sequence is therefore an infinite sequence of independent pairs $(x_1, z_1), (x_2, z_2), \ldots, (x_k, z_k), \ldots$. The decision rule that minimizes the probability of misclassification can be expressed in terms of the Bayesian *a posteriori* probabilities:

$$\begin{aligned} x \in \omega_1, &\quad \text{if} \quad Pr\{\omega_1|x\} - Pr\{\omega_2|x\} > 0; \\ x \in \omega_2, &\quad \text{if} \quad Pr\{\omega_1|x\} - Pr\{\omega_2|x\} < 0. \end{aligned}$$

Since the $Pr\{\omega_i|x\}$ are not known, one can attempt to find a parameter vector $\theta \in \Re^n$ such that $\theta^T x$ best approximates $Pr\{\omega_1|x\} - Pr\{\omega_2|x\}$, where $\theta^T x$ is the inner product of θ and x, and then use the decision rule

$$\begin{aligned} x \in \omega_1, &\quad \text{if} \quad \theta^T x > 0; \\ x \in \omega_2, &\quad \text{if} \quad \theta^T x < 0. \end{aligned}$$

It can be shown (e.g., ref [26]) that the mean-square approximation error is minimized by finding the θ that minimizes the functional

$$J(\theta) = E\{(\theta^T x - z)^2\}.$$

The functional $J(\theta)$ is not known explicitly, but each step k in the training procedure provides the opportunity to observe a sample error $(\theta_k^T x_k - z_k)^2$. The derivative of this sample error with respect to θ_k can be regarded as a noisy measure of the gradient of $J(\theta)$ with respect to θ:

$$y(\theta_k) = 2[\theta_k^T x_k - z_k]x_k,$$

for which it is true that

$$E\{y(\theta)|\theta\} = \nabla_\theta J(\theta).$$

In this case, $\nabla_\theta J(\theta)$ has a unique zero at some value $\theta = \theta^\circ$. The problem is therefore one of finding the zero of the unknown function $r(\theta) = \nabla_\theta J(\theta)$ by using the noisy measurements $y(\theta)$. The Robbins-Monro algorithm can therefore be applied. This is a gradient-descent procedure that uses the random variable $y(\theta)$ in place of the actual, but unobservable, values of the gradient $r(\theta)$:

$$\theta_{k+1} = \theta_k - \rho_k y(\theta_k). \tag{4}$$

The following assumptions, from [4], are sufficient for the convergence result stated below:[2]

(A1): $r(\theta) = E\{y(\theta)|\theta\}$ has a unique zero at $\theta = \theta^\circ$.

(A2): The sequence ρ_k has the following properties:

$$\rho_k \geq 0, \quad \sum_k \rho_k = \infty, \quad \sum_k \rho_k^2 < \infty.$$

(A3): $r(\theta)$ behaves like a linear function for values of θ near θ°. That is

$$\inf_{\epsilon<\|\theta-\theta^\circ\|<\epsilon^{-1}} |(\theta - \theta^\circ)^{\mathrm{T}} r(\theta)| > 0, \quad \forall \epsilon > 0.$$

(A4):

$$E\{\|y(\theta)\|^2\} \leq h(1 + \|\theta - \theta^\circ\|^2), \quad h > 0.$$

If assumptions (A1)-(A4) are satisfied, then the algorithm (4) converges to the unique value θ° in the sense that

$$\lim_{k\to\infty} E\{\|\theta_k - \theta^\circ\|^2\} = 0$$

and

$$Pr\{\lim_{k\to\infty} \theta_k = \theta^\circ\} = 1.$$

[2] Results for a variety of less restrictive conditions have also been proven. See, for example, the review [4]. However, our proof refers to the conditions stated here.

Applied to the pattern classification problem, the Robbins-Monro algorithm takes the form

$$\theta_{k+1} = \theta_k - \rho_k[\theta_k^T x_k - z_k]x_k. \tag{5}$$

This algorithm can be related to the pseudoinverse of a linear operator and to regression analysis [29]. A similar algorithm was presented by Widrow and Hoff [30] in the form of a neuron-like adaptive element which they called an *adaline* (for *ada*ptive *lin*ear *e*lement).

We now define an *associative reinforcement learning task*, in which each class label of a supervised learning pattern-classification task is replaced by a success/failure signal that depends probabilistically upon the action of the learning system and the pattern vector. Then we discuss the distinction between this success/failure information and the information provided by class labels.

IV. ASSOCIATIVE REINFORCEMENT LEARNING

In an associative reinforcement learning task, as we define it here, the learning system and its environment interact in a closed loop. At time step k the environment provides the learning system with a random pattern vector, x_k, selected independently from a finite set $X = \{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \Re^n$; the learning system emits an action, a_k, chosen from a finite set $A = \{a^{(1)}, \ldots, a^{(r)}\}$; the environment receives a_k as input and sends to the learning system a success/failure signal $b_k \in B = \{1, -1\}$ that evaluates the action a_k. The environment determines the evaluation according to a map $d : X \times A \rightarrow [0,1]$, where $d(x,a) = Pr\{b_k = 1 \mid x_k = x, a_k = a\}$. Ideally, one wants the learning system eventually to respond to each input pattern $x \in X$ with the action a^x with probability 1, where a^x is such that $d(x, a^x) = \max_{a \in A}\{d(x,a)\}$.

Clearly, in the case of a single input pattern ($|X| = 1$), this task reduces to the learning automaton task described above. On the other hand, in the case of two actions ($|A| = 2$) the task reduces to the supervised learning pattern-classification task described in Section III if for each $x \in X$, $d(x, a^{(1)}) + d(x, a^{(2)}) = 1$. Specifically, encode the actions as $a^{(1)} = 1$, $a^{(2)} = -1$, and define the class label to be $z_k = b_k a_k$, thus yielding the class label 1 when action $a_k = 1$ yields success ($b_k = 1$) or when action $a_k = -1$ yields failure ($b_k = -1$). Similarly, the class label is -1 when action $a_k = 1$ yields failure or when action $a_k = -1$ yields success. In other words, upon success, assume that the class label provided by the teacher equals the action actually taken; upon failure, assume that the class label equals the action *not* taken (this is what Widrow *et al.* mean by "selective bootstrap adaptation").

Then

$$\begin{aligned} Pr\{z_k = 1|x_k = x\} &= Pr\{b_k = 1, a_k = 1|x_k = x\} + Pr\{b_k = -1, a_k = -1|x_k = x\} \\ &= d(x,1)Pr\{a_k = 1|x_k = x\} + [1 - d(x,-1)]Pr\{a_k = -1|x_k = x\}, \end{aligned} \tag{6a}$$

and similarly

$$Pr\{z_k = -1|x_k = x\} = d(x,-1)Pr\{a_k = -1|x_k = x\} + [1 - d(x,1)]Pr\{a_k = 1|x_k = x\}. \tag{6b}$$

In the supervised learning pattern-classification task, the probabilities given by (6a) and (6b), which respectively equal $Pr\{\omega_1|x\}$ and $Pr\{\omega_2|x\}$, are independent of the learning system actions. This only occurs if for each $x \in X$, $d(x,a^{(1)}) + d(x,a^{(2)}) = 1$, in which case

$$Pr\{z_k = 1|x_k = x\} = d(x,1),$$

$$Pr\{z_k = -1|x_k = x\} = d(x,-1).$$

Under these conditions, a solution to the associative reinforcement learning task is a solution to the corresponding supervised learning pattern-classification task.

This reduction of associative reinforcement learning to supervised learning pattern-classification requires the restriction to two actions since it must be possible to estimate, on the basis of the action taken and the success/failure feedback, what the action should have been (i.e., to estimate the class label of the input pattern). In the case of three or more actions, a success/failure signal is clearly not as informative as a signal providing a class label since evaluative feedback only provides information about the action actually taken and not about any of the other actions. This remains true in the case of two actions because an associative reinforcement learning task involving actions $a^{(1)}$ and $a^{(2)}$ does not require that $d(x,a^{(1)}) + d(x,a^{(2)}) = 1$ for all $x \in X$. It may be the case that for some patterns the optimal action yields failure most of the time, or that the nonoptimal action yields success most of the time. The frequency of success produced by one action provides no information about about the frequency of success produced by the other action. Appendix I contains additional observations regarding this distinction and its implications for algorithms.

Thus, even in the case of two actions, there is a deep distinction between associative reinforcement learning and supervised learning pattern-classification. Supervised learning pattern classification is more of a "rote" form of learning than is associative reinforcement learning. In the former task, the system need not *discover* which actions are optimal by testing all actions and settling on the one that yields the highest evaluation, as a learning automaton must do in the nonassociative case, and as is required in associative reinforcement learning. Various extensions of the task defined here are of obvious interest but are not discussed in the present article: Infinite pattern sets, statistical dependence among the input patterns, dependency of input patterns on previous actions of the learning system, delayed evaluation, etc. Extensions that allow control of pattern input as well as evaluative input make contact with Markovian decision theory [31], [32] and studies of learning automata for controlling Markov processes [1], [24]. In these latter studies, however, the pattern-classification aspects of our problem are not considered. Simulation studies of associative reinforcement learning involving pattern classification, control, and delayed evaluation are reported by Sutton [33].

V. THE A_{R-P} ALGORITHM

The algorithm we have studied is a combination of a stochastic learning automaton algorithm and the pattern-classification algorithm based on stochastic approximation (5). It is restricted to the case of two actions, and like the stochastic approximation algorithm, its decision rule is parameterized at step k by a vector $\theta_k \in \Re^n$. Unlike that algorithm, however, its decision is not a deterministic function of the input pattern:

$$a_k = \begin{cases} 1, & \text{if } \theta_k^T x_k + \eta_k > 0; \\ -1, & \text{otherwise;} \end{cases} \tag{7}$$

where the η_k are independent, identically distributed random variables, each having distribution function Ψ. The parameter vector is updated according to the following equation:

$$\theta_{k+1} - \theta_k = \begin{cases} -\rho_k[E\{a_k|\theta_k, x_k\} - b_k a_k]x_k, & \text{if } b_k = 1 \text{ (success);} \\ -\lambda\rho_k[E\{a_k|\theta_k, x_k\} - b_k a_k]x_k, & \text{if } b_k = -1 \text{ (failure);} \end{cases} \tag{8}$$

where $0 \leq \lambda \leq 1$ and $\rho_k > 0$. If $\lambda \neq 0$, we call this the *associative reward-penalty* (A_{R-P}) algorithm, and if $\lambda = 0$, we call it the *associative reward-inaction* (A_{R-I}) algorithm. Since the distribution Ψ is a part of the algorithm rather than a part of the environment, it is assumed to be known. Consequently the expected value in (8) can be determined as a known function of θ_k and x_k, and thus does not have to be estimated (see below for a special case). We have retained the product $b_k a_k$ in (8) rather than substituting the values for b_k in order to emphasize its similarity to (5).

According to this algorithm, the action probabilities are conditional on the input pattern in a manner determined by the parameter vector θ_k. If $E\{\eta_k\} = 0$, then when $\theta_k^T x_k = 0$ the probability that each action is emitted, given input pattern x_k, is .5; when $\theta_k^T x_k$ is positive, action $a_k = 1$ is the more likely action; and when $\theta_k^T x_k$ is negative, action $a_k = -1$ is the more likely. As θ changes according to (8), the mapping that determines the action probabilities as a function of the input pattern changes. According to (8), θ changes so as to reduce the discrepancy between the action expectation, $E\{a_k|\theta_k, x_k\}$, for input pattern x_k, and the estimated correct action, $b_k a_k$, corresponding to x_k.

Just as the associative reinforcement learning task reduces to examples of both the learning automaton task and the supervised learning pattern-classification task under differents sets of restrictions, this algorithm reduces to corresponding algorithms under certain sets of restrictions. We carry out this reduction in some detail since it makes the nature of the algorithm and its relationship to existing algorithms clear. We first show how it reduces to the two-action L_{R-P} algorithm.

Let each random variable η_k be uniform in the interval $[-1, 1]$ so that

$$\Psi(r) = Pr\{\eta_k \leq r\} = \begin{cases} 0, & \text{if } r \leq -1; \\ (1 + r)/2, & \text{if } -1 < r < 1; \\ 1, & \text{if } r \geq 1. \end{cases} \tag{9}$$

Then

$$p_k^{(-1)} = Pr\{a_k = -1\} = Pr\{\theta_k^{\mathrm{T}} x_k + \eta_k \le 0\}$$
$$= \Psi(-\theta_k^{\mathrm{T}} x_k),$$

and

$$p_k^{(+1)} = Pr\{a_k = +1\} = 1 - \Psi(-\theta_k^{\mathrm{T}} x_k).$$

We can write the expected value used in (8) as follows:

$$\begin{aligned} E\{a_k|\theta_k, x_k\} &= -1 \cdot p_k^{(-1)} + 1 \cdot p_k^{(+1)} \\ &= 1 - 2\Psi(-\theta_k^{\mathrm{T}} x_k) \\ &= \tau(\theta_k^{\mathrm{T}} x_k), \end{aligned}$$

where

$$\tau(r) = \begin{cases} -1, & \text{if } r \le -1; \\ r, & \text{if } -1 < r < 1; \\ 1, & \text{if } r \ge 1. \end{cases}$$

Further, suppose that the input vector remains constant (and nonzero) over time, i.e., $x_k = \hat{x} \ne 0$, $k \ge 1$. Then (8) can be written

$$\theta_{k+1} - \theta_k = \begin{cases} -\rho_k[\tau(\theta_k^{\mathrm{T}} \hat{x}) - b_k a_k]\hat{x}, & \text{if } b_k = 1 \text{ (success)}; \\ -\lambda\rho_k[\tau(\theta_k^{\mathrm{T}} \hat{x}) - b_k a_k]\hat{x}, & \text{if } b_k = -1 \text{ (failure)}. \end{cases}$$

Assume $\theta_k^{\mathrm{T}} \hat{x} \in [-1, 1]$ and consider the case in which $a_k = 1$ and $b_k = 1$. Then

$$\begin{aligned} p_{k+1}^{(-1)} &= \Psi(-\theta_{k+1}^{\mathrm{T}} \hat{x}) \\ &= \Psi\Big((-\theta_k + \rho_k[\tau(\theta_k^{\mathrm{T}} \hat{x}) - 1]\hat{x})^{\mathrm{T}} \hat{x}\Big) \\ &= \Psi\Big(-\theta_k^{\mathrm{T}} \hat{x} + \rho_k \|\hat{x}\|^2 [\theta_k^{\mathrm{T}} \hat{x} - 1]\Big). \end{aligned} \tag{10}$$

If we assume that ρ_k is chosen so that $\rho_k \|\hat{x}\|^2 \in [0, 1]$, then the argument of Ψ in (10) is a convex combination of -1 and $-\theta_k^{\mathrm{T}} \hat{x}$ and thus is in the interval $[-1, -\theta_k^{\mathrm{T}} \hat{x}] \subseteq [-1, 1]$.

Consequently, according to (9) we have that

$$\begin{aligned} p_{k+1}^{(-1)} &= (1 - \theta_k^T\hat{x} + \rho_k\|\hat{x}\|^2[\theta_k^T\hat{x} - 1])/2 \\ &= (1 - \rho_k\|\hat{x}\|^2)(1 - \theta_k^T\hat{x})/2 \\ &= (1 - \rho_k\|\hat{x}\|^2)\Psi(-\theta_k^T\hat{x}) \\ &= (1 - \rho_k\|\hat{x}\|^2)p_k^{(-1)}. \end{aligned}$$

Thus, assuming that $\theta_k^T\hat{x} \in [-1,1]$, we can choose $\rho_k \equiv \rho$ small enough to make $\rho\|\hat{x}\|^2 \in [0,1]$ so that in the case of success, the algorithm reduces to the two-action L_{R-P} algorithm with $\alpha = \rho\|\hat{x}\|^2$ (Eq. 1). A similar argument shows that in the case of failure, if $\theta_k^T\hat{x} \in [-1,1]$, the algorithm reduces to the two-action L_{R-P} scheme with $\beta = \lambda\rho\|\hat{x}\|^2$ (Eq. 2). In case $\theta_k^T\hat{x} \notin [-1,1]$, it can happen that the algorithm is not equivalent to the L_{R-P} algorithm. However, if we require that at the initial step $k = 1$, $\theta_1^T\hat{x} \in [-1,1]$, then it is clear that $\theta_k^T\hat{x} \in [-1,1]$ for all $k > 1$. Summarizing, the A_{R-P} algorithm reduces to the two-action (nonassociative) L_{R-P} algorithm if 1) each η_k is uniform in the interval $[-1,1]$; 2) the input pattern is constant and nonzero over time ($\equiv \hat{x}$); and 3) $\theta_1^T\hat{x} \in [-1,1]$. Obviously, the A_{R-I} algorithm reduces to the two-action (nonassociative) L_{R-I} algorithm under the same restrictions. If the random variables η_k are not uniform, then the A_{R-P} algorithm similarly reduces to what Lakshmivarahan [2] calls a nonlinear reward-penalty algorithm of the non-absorbing type, of which the L_{R-P} algorithm is a special case.

On the other hand, the A_{R-P} algorithm transforms to the two-category supervised learning pattern-classification algorithm (5) as follows. Let Ψ be the step function

$$\Psi(r) = \begin{cases} 0, & \text{if } r < 0; \\ 1, & \text{if } r \geq 0; \end{cases}$$

meaning that $\eta_k = 0$ for all $k \geq 1$. Then according to (7) each action a_k is computed from $\theta_k^T x_k$ by comparison with a deterministic threshold of zero. Let us regard a_k as the "external" action that is evaluated by the environment, but let us regard the scalar $\theta_k^T x_k$ as an "internal" action that is used to determine the expected value in (8). Then $E\{a_k|\theta_k, x_k\} = \theta_k^T x_k$, and if one additionally lets $\lambda = 1$ and $z_k = b_k a_k$, (8) becomes identical to (5). This latter computation, which permits the algorithm to accept success/failure feedback, b_k, rather than direct input of class labels, z_k, is usually not included in supervised learning pattern-classification algorithms, but it is a simple yet important modification proposed by Widrow *et al.* [12] to form the "selective bootstrap adaptation" algorithm, as they call this deterministic algorithm. Since the "internal" and "external" actions are different, this algorithm is not an exact specialization of the A_{R-P} algorithm, and our convergence theorem does not apply to it. Simulation results (see Section VII) suggest that it does not perform well in all associative reinforcement learning tasks.

The A_{R-P} algorithm does, however, exactly specialize to the perceptron learning algorithm [3], [26], by letting Ψ be the same step function as above and letting both the "internal" and "external" actions be a_k as computed by (7) with the resulting deterministic threshold. Then

$$E\{a_k|\theta_k, x_k\} = a_k = \begin{cases} 1, & \text{if } \theta_k^{\mathrm{T}} x_k > 0; \\ -1, & \text{otherwise,} \end{cases}$$

which when substituted into (8) with $\lambda = 1$ and $b_k a_k$ replaced by the externally supplied class label z_k, yields the perceptron algorithm. Note that in this case the parameter vector, θ_k, is updated only in the case of failure. Hence, the perceptron algorithm (extended to accept success/failure signals) can be viewed as an extension to the associative case of a deterministic inaction-penalty learning automaton algorithm, a type of algorithm that has severe difficulty in any stochastic environment. Our convergence result does not apply to this specialization of the A_{R-P} algorithm since this Ψ is not continuous and strictly monotonic.

Finally, note that if the set X of input patterns consists of standard-unit-basis vectors $x^{(i)} = (0, \ldots, 1, \ldots, 0)^{\mathrm{T}}$, then the A_{R-P} algorithm reduces to the lookup-table method of making learning automata sensitive to environmental state. Each input vector addresses a separate parameter that determines the action probabilities for that case, and no generalization occurs. Barto *et al.* [8] used an algorithm similar to the A_{R-P} algorithm under this restriction in a control problem.

VI. CONVERGENCE THEOREM

The following theorem is an extension of a strong convergence result proven by Lakshmivarahan (Theorem 4.1 in ref. [1], p. 112) for a class of time-varying learning automaton algorithms, and we use notation similar to his. (These algorithms are called time-varying because the parameter sequence ρ_k is not constant.) We extend this result to the A_{R-P} algorithm applied to associative reinforcement learning tasks in which the set X of input patterns is a linearly independent set. The extension of the result to the case of arbitrary sets X is currently under examination.

We delineate the following conditions on the associative reinforcement learning task and on the A_{R-P} algorithm:

(C1): The set of input vectors $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ is a linearly independent set.

(C2): For each $x^{(i)} \in X$ and $k \geq 1$, $Pr\{x_k = x^{(i)}\} = \xi^i > 0$.

(C3): Each of the independent, identically distributed random variables η_k in (7) have a continuous and strictly monotonic distribution function Ψ.

(C4): The sequence ρ_k in (8) satisfies (A2) of Section III; that is

$$\rho_k \geq 0, \quad \sum_k \rho_k = \infty, \quad \sum_k \rho_k^2 < \infty.$$

We can prove the following theorem:

Theorem. *Under conditions (C1)-(C4), for each $\lambda \in (0,1]$, there exists a $\theta_\lambda^\circ \in \Re^n$ such that the random process $\{\theta_k\}_{k\geq 1}$ generated by the A_{R-P} algorithm in an associative reinforcement learning task converges to θ_λ° with probability 1 (that is, $Pr\{\lim_{k\to\infty} \theta_k = \theta_\lambda^\circ\} = 1$), where for all $x \in X$,*

$$\begin{aligned} Pr\{a = 1 | \theta_\lambda^\circ, x\} &> .5, \quad \text{if} \quad d(x,1) > d(x,-1); \\ &< .5, \quad \text{if} \quad d(x,1) < d(x,-1). \end{aligned}$$

In addition, for all $x \in X$,

$$\lim_{\lambda\to 0} Pr\{a = 1 | \theta_\lambda^\circ, x\} = \begin{cases} 1, & \textit{if } d(x,1) > d(x,-1); \\ 0, & \textit{if } d(x,1) < d(x,-1). \end{cases}$$

According to the usual performance criteria for learning automata defined above, this result implies that for each $x \in X$, the A_{R-P} algorithm is ϵ-optimal. In fact, it implies a strong form of ϵ-optimality for each $x \in X$ in which individual runs of the algorithm will (almost surely) converge so as to produce the optimal action for each pattern vector with a probability that can be made as close to one as desired by adjusting the parameter λ. See Appendix II for a proof.

Remarks

1) We present this strong convergence theorem because we found Lakshmivarahan's [1] proof for the nonassociative time-varying case the easiest to extend. A variety of results with less-restrictive conditions are undoubtedly possible, and a variety of proof techniques are applicable. For example, simulations suggest that the usual form of ϵ-optimality (Section II) may hold for each $x \in X$ for the A_{R-P} algorithm with the parameter sequence ρ_k held constant over k. An extension of Lakshmivarahan's proof (refs. [1] and [2], Chapter 2) based on the work of Norman [34], [35], would apply to this case, but we have not worked it out. We have also not explored the application of weak convergence methods. Finally, it is likely that condition (C1), which requires linear independence of pattern vectors, can be removed and a result can be proved concerning the best-least-squares approximation to optimal performance analogous to the stochastic approximation result described in Section III. However, we do not yet understand how the A_{R-P} algorithm behaves when the pattern vectors are dependent and are continuing to examine this case.

2) Note that since this theorem only applies to cases in which the distribution finction Ψ is strictly monotonic, it does not apply to the case in which the random variables η_k are uniform in the interval $[-1, 1]$. Thus, it does not apply to the version of the A_{R-P} algorithm that specializes to the nonassociative L_{R-P} algorithm in the manner described in Section V. A modified version of the proof, however, does apply to the case in which the η_k are uniform random variables.

3) Also since the theorem requires the distribution function Ψ to be strictly monotonic, it does not apply to the case in which the A_{R-P} algorithm reduces to the perceptron algorithm, that is, the case in which Ψ is the step function. The result is in fact false in

this case since when its pattern input is held constant, the perceptron algorithm (modified to accept success/failure signals) becomes the two-state Tsetlin automaton, an algorithm which performs poorly in all stochastic environments [19], [36].

4) Simulation experiments suggest that the theorem is not true for $\lambda = 0$; that is, the A_{R-I} algorithm does not converge to the correct actions for all linearly independent sets of input patterns. It appears that the parameter vector θ must be adjusted in cases of failure in order to discriminate arbitrary linearly independent input vectors. This is suggested by the form of conventional pattern-classification algorithms of the error-correcting type which, going to the other extreme, update the parameter vector only upon error.

5) It is not immediately obvious how algorithms, like the A_{R-P} algorithm, that determine their actions via a noisy threshold can be generalised to the case of more than two actions. The most promising avenue for extension may parallel that used in extending algorithms for supervised learning pattern classification to the n-class case. In this scheme, the classification system consists of a bank of n devices, each receiving identical input patterns. For each pattern, each device determines the value of a discriminant function based on its current parameter values. The decision is made by determining which discriminant function has the maximum value [26]. An n-action extension of the A_{R-P} algorithm is obtained if each device in this organization implements the A_{R-P} algorithm, with $E\{a_k|\theta_k, x_k\}$ playing the role of discriminant function. We have not yet studied this type of system in depth.

VII. SIMULATION RESULTS

We simulated the A_{R-P} algorithm and the selective bootstrap adaptation algorithm of Widrow *et al.* [12] in two simple associative reinforcement learning tasks. Our goal was to illustrate the convergence theorem, rate of convergence, and the limitations of the bootstrap algorithm as a representative of the class of conventional pattern-classification algorithms. Unlike the nonassociative tasks in which learning automata are studied, associative tasks can be very easy or very difficult to solve depending on the dimensionality of the input patterns, their number, and the geometry of the required classification. Associative reinforcement learning has the additional degree of variation provided by the possibilities for selecting the success probabilities for each input pattern. Here we have chosen tasks that are as simple as possible while still making apparent the important distinctions between various classes of algorithms. Obviously questions arise, especially about convergence rate, that are not answered by these minimal simulations.

We measure performance by computing at each step, k, the expected probability of success taken over the two actions and all of the input patterns. Letting $p_k^{1i} = Pr\{a_k = 1 | x_k = x^{(i)}\}$ and $p_k^{-1i} = Pr\{a_k = -1 | x_k = x^{(i)}\} = 1 - p_k^{1i}$, this is

$$\begin{aligned} M_k &= \sum_{i=1}^{m} \xi^i [Pr\{b_k = 1 | x_k = x^{(i)}\}] \\ &= \sum_{i=1}^{m} \xi^i [d(x^{(i)}, 1) p_k^{1i} + d(x^{(i)}, -1) p_k^{-1i}]. \end{aligned}$$

This measure is the generalization to the associative case of the measure defined by (3) for the nonassociative case. It is maximized when the optimal action for each input pattern occurs with probability 1, in which case it is

$$M_{\max} = \sum_{i=1}^{m} \xi^i \max[d(x^{(i)}, 1), d(x^{(i)}, -1)].$$

The distribution function Ψ that we use in each simulation is the logistic distribution given by

$$\Psi(r) = \frac{1}{1 + e^{-r/T}},$$

where T is a parameter. This function is easily computed, as is its inverse, and it satisfies condition (C3) of the theorem.[3] We set $T = .5$ in all simulations so that the slope of $E\{a_k | \theta_k, x_k\}$ as a function of $\theta_k^T x_k$ is 1 for $\theta_k^T x_k = 0$.

Each task requires a discrimination to be made between two input vectors: $x^{(1)} = (1,1)^T$ and $x^{(2)} = (1,0)^T$, which are linearly independent but not orthogonal. These vectors are equally likely to occur at each time step ($\xi^1 = \xi^2 = .5$). We set the initial

[3] This distribution arises in stochastic relaxation methods that involve flipping binary states in models having having their origin in statistical thermodynamics. It is used by Hinton and colleagues [37],[38], and Smolensky [39] who apply these methods to problems in Artificial Intelligence. The parameter T corresponds to the temperature of a thermodynamic system.

parameter vector, θ_1, equal to the zero vector for each run, which makes the actions initially equiprobable for all input patterns. We did not systematically search the parameter space for optimum values of ρ_k and λ for the tasks presented, but we discuss the effect of parameter choice on the comparisons we make between algorithms where it is important.

A. Task 1

For Task 1 the map $d : X \times A \mapsto [0,1]$ giving the success probabilities for each action and each input pattern is:

$$d(x^{(1)}, -1) = .8$$
$$d(x^{(1)}, +1) = .1$$
$$d(x^{(2)}, -1) = .2$$
$$d(x^{(2)}, +1) = .9.$$

Thus it is optimal for the learning system to respond to $x^{(1)}$ and $x^{(2)}$ with the actions -1 and $+1$ respectively. In this case $M_{\max} = .85$. The initial expected success probability is .5. Even though this task is not reducible to a supervised learning pattern-classification task since $d(x^{(i)}, 1) + d(x^{(i)}, -1) \neq 1$, $i = 1, 2$, it is relatively easy since the success probabilities for each input pattern are widely separated and are not both greater than, or both less than, .5 (case i of Appendix I).

Fig. 1a shows results of simulating the A_{R-P} algorithm for three values of λ, the selective bootstrap algorithm of Widrow *et al.* [12], and the nonassociative L_{R-P} algorithm. Plotted for each time step k, $1 \leq k \leq 1{,}000$, is the average of M_k over 100 runs for each algorithm and parameter setting. We held the parameter ρ_k at the value .5 for all k in the A_{R-P} algorithm, so the strong convergence result is not illustrated by the data in Fig. 1a. The fastest rising curve is that produced by the bootstrap algorithm ($\rho_k = .1$ for all k, $\lambda = 1.0$) showing its effectiveness when success probabilities for each input pattern are placed on either side of .5 (Task 2 makes evident the advantage of the A_{R-P} algorithm over the selective bootstrap algorithm for more difficult problems). The curve at the bottom that identically equals .5 is the performance of the nonassociative L_{R-P} algorithm, and in fact, would be the performance of *any* nonassociative learning automaton algorithm with *any* parameter setting. Since these algorithms cannot detect the input vectors, and therefore cannot discriminate between them, $M_k = .5$ for all action probabilities.

The remaining three curves show the performance of the A_{R-P} algorithm. The dashed lines show the theoretical asymptotic values for M_k of .8485, .8152, and .7868 for λ respectively .01, .25, and .5. These data show that the average over the 100 runs of the expected success probability comes to remain fairly close to these theoretical values.[4] Note that as in the case of nonassociative algorithms, the learning rate slows as λ decreases (and the asymptote correspondingly increases).

Fig. 1b is a plot of M_k, $1 \leq k \leq 10{,}000$, for a single run of the A_{R-P} algorithm with $\lambda = .25$ and $\rho_k = 1/k^{.55}$. This ρ-sequence satisfies condition (C4) of the theorem. The expected success probability after 10,000 steps is .8163, which closely approximates the theoretical asymptotic value .8152. The parameter vector θ_k after 10,000 steps is $(1.648, -3.018)^T$, which is close to the theoretical asymptotic value $(1.671, -3.002)^T$. This realization of the stochastic process therefore appears to to act as it should according to the strong convergence theorem. Not surprisingly, decreasing ρ_k considerably slows the learning process.

B. Task 2

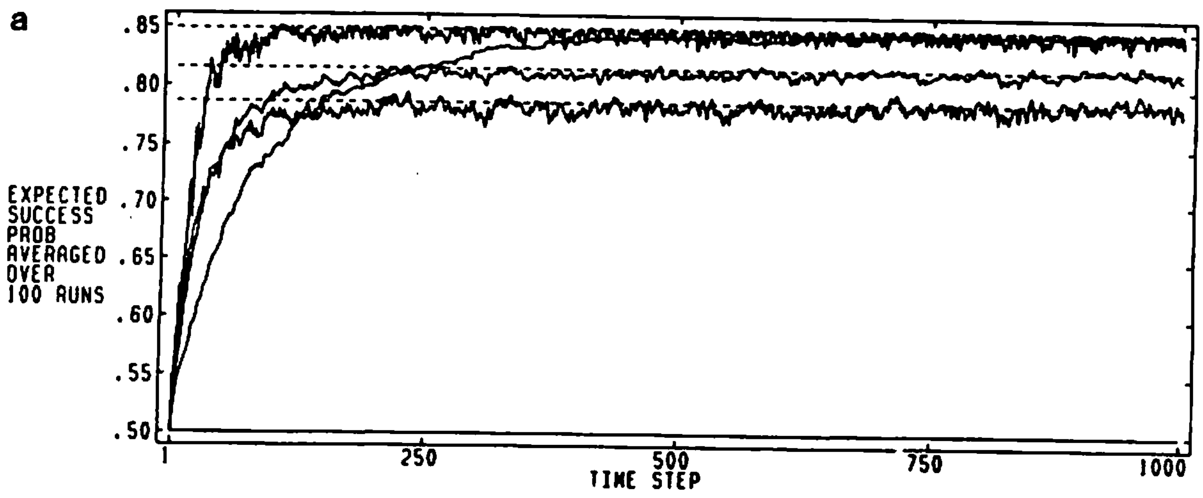
For Task 2 the map $d : X \times A \mapsto [0,1]$ giving the success probabilities for each action and each input pattern is:

$$d(x^{(1)}, -1) = .2$$
$$d(x^{(1)}, +1) = .4$$
$$d(x^{(2)}, -1) = .9$$
$$d(x^{(2)}, +1) = .6.$$

Here it is optimal for the algorithm to respond to $x^{(1)}$ and $x^{(2)}$ with the actions $+1$ and -1 respectively. In this case $M_{\max} = .65$, and the initial expected success probability,

[4] These theoretical values can be computed by referring to the proof in Appendix II. Each value β^i_λ, $1 \leq i \leq m$ in (13) is determined by solving for $p^{1i}(\theta)$ in the equation $w^i(\theta, \lambda) = 0$, which is quadratic in $p^{1i}(\theta)$. These values are used to compute the vector $\bar{\beta}_\lambda$ via the inverse of Ψ. From this can be determined θ°_λ and the asymptotic value of M_k.

a



b

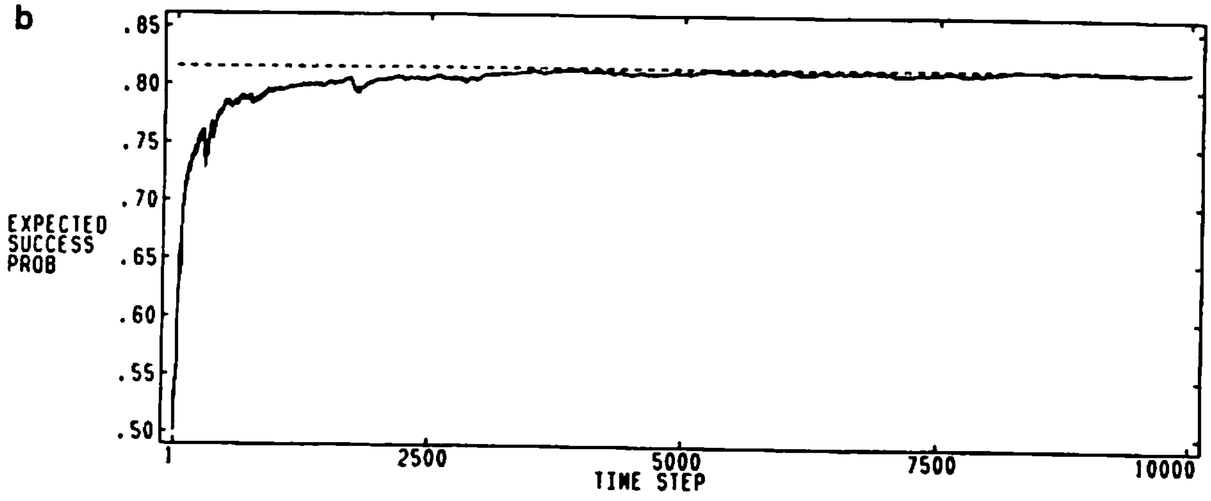


Figure 1

Simulation results for Task 1. (a) Curves showing the averages of M_k over 100 runs of the selective bootstrap algorithm (the fastest rising curve) and the A_{R-P} algorithm for three values of λ. The sequence ρ_k is constant in all cases. (b) Curve showing M_k for a single run of the A_{R-P} algorithm with ρ_k decreasing with increasing k.

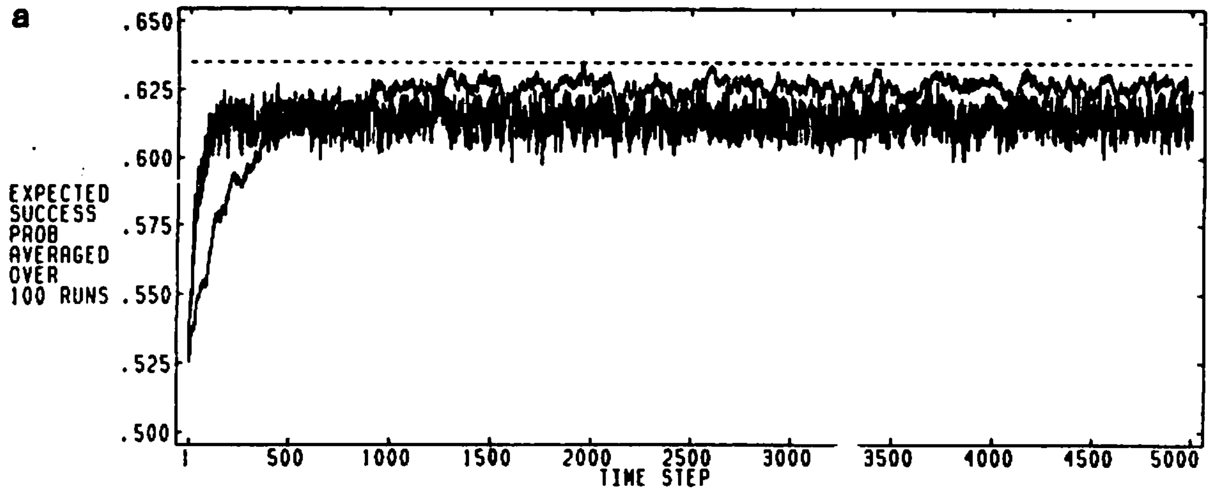
M_1, is .525. Since the success probabilities are both less than .5 for pattern $x^{(1)}$ (case iii of Appendix I) and both greater than .5 for pattern $x^{(2)}$ (case ii of Appendix I), this task is considerably more difficult than Task 1. An algorithm that responds correctly to pattern $x^{(1)}$ but incorrectly to pattern $x^{(2)}$ will have an expected success probability of .5 rather than the optimal value .65. On the other hand, an algorithm that responds correctly to pattern $x^{(2)}$ but incorrectly to pattern $x^{(1)}$ will have an expected success probability of .55.

Fig. 2a shows results of simulating the A_{R-P} algorithm with $\lambda = .05$ and $\rho_k = .5$ for all k, and the selective bootstrap algorithm of Widrow *et al.* [12] with $\lambda = 1.0$ and $\rho_k = .1$ for all k (the same parameter values as in Task 1). Plotted for each time step, k, $1 \leq k \leq 5{,}000$, is the average of M_k over 100 runs for each algorithm. The dashed line shows the theoretical asymptotic value for M_k of .6348 for the A_{R-P} algorithm, but here ρ_k is constant so the conditions for strong convergence are not in force. The curve closer to this asymptote is due to the A_{R-P} algorithm; the other curve is due to the selective bootstrap algorithm. Fig. 2b shows results obtained under conditions identical to those that produced Fig. 2a except that ρ_k for the A_{R-P} algorithm is .1 for all k instead of .5. Here, the average of M_k approaches the asymptote more closely. This would be in accord with an extension of the result of Lakshmivarahan [1], [2], for the nonassociative case with ρ_k held constant over each run, in which the $lim_{k \to \infty} E\{M_k\}$ is shown to approach the asymptote more closely for smaller constant values of ρ_k.[5] We have not yet extended this result to the associative case.

Fig. 3a is a plot of M_k, $1 \leq k \leq 1{,}000$, for a single run of the selective bootstrap algorithm with the parameters used to produce Fig. 2. The algorithm very quickly produces the correct action for pattern $x^{(2)}$ but continues to oscillate between the correct and incorrect actions for pattern $x^{(1)}$, spending slighly more time with the correct action. This oscillation persisted as long as we observed the algorithm's behavior, and all runs at this parameter setting that we observed showed this behavior. Thus the average for the selective bootstrap algorithm shown in Fig. 2 is the average of runs showing this oscillatory behavior. Fig. 3b is a plot of a single run of the A_{R-P} algorithm with $\lambda = .05$, $\rho_1 = 2.0$, and $\rho_k = 1/k^{.55}$. Thus the conditions of the theorem are satisfied, and this realization of the stochastic process appears to be acting appropriately.

[5] This is not to be confused with the strong convergence result extended in the present article in which it is M_k, and not $E(M_k)$, that approaches the asymptote with probabilty 1 if the sequence ρ_k decreases at an appropriate rate with increasing k.

a



b

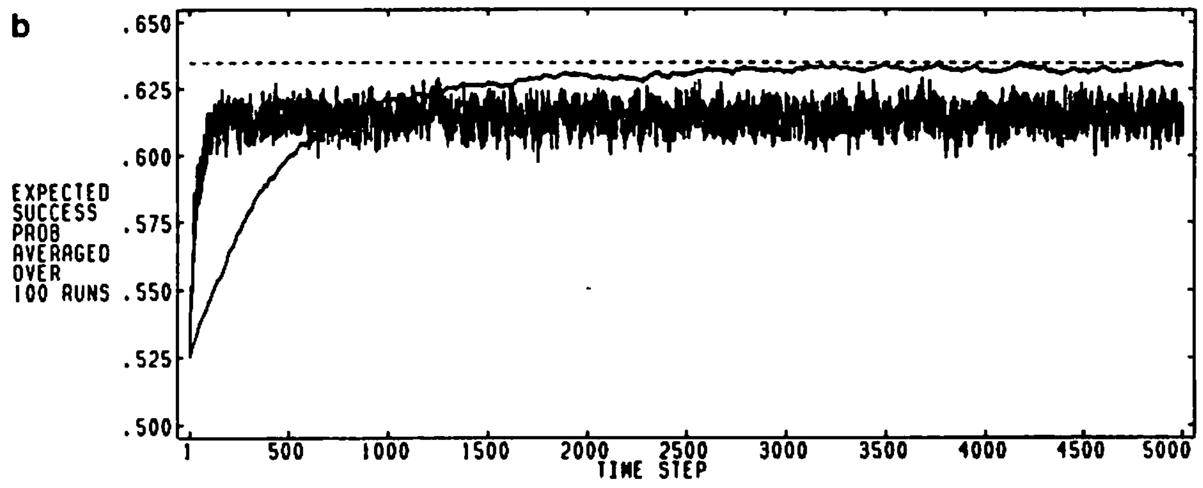


Figure 2

Simulation results for Task 2 averaged over 100 runs with constant parameter sequence ρ_k. (a) Curves showing the averages of M_k over 100 runs of the selective bootstrap algorithm (the lower curve) and the A_{R-P} algorithm. (b) Curves produced under conditions identical to those of (a) except the A_{R-P} algorithm uses a smaller constant value of ρ_k.

The oscillatory behavior of the selective bootstrap algorithm in this task can be eliminated with another choice for the value of λ. For example, with $\lambda = .125$, in some runs the oscillation is transient and the correct solution stably forms. In other runs at this parameter setting, however, the algorithm converges to the wrong action for pattern $x^{(2)}$

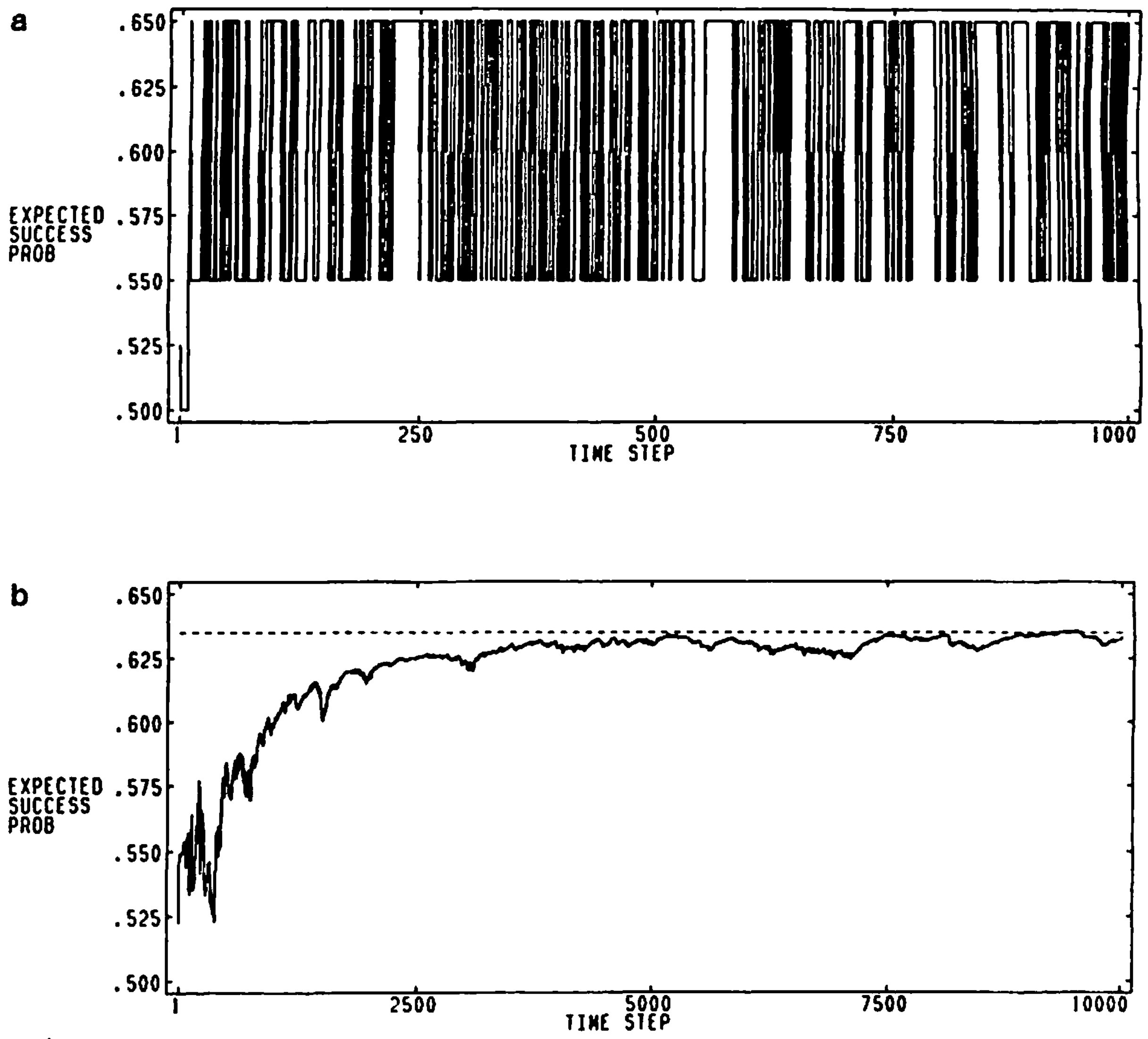


Figure 3

Simulation results for single runs of Task 2. (a) A single run of the selective bootstrap algorithm. (b) A single run of the A_{R-P} algorithm with decreasing sequence ρ_k.

so that M_k remains at .5. A value for λ less than 1 compensates for the fact that both success probabilities for pattern $x^{(1)}$ are less than .5. Similarly, setting λ greater than 1 can compensate for success probabilities that are both greater than .5. Therefore, given enough *a priori* knowledge about the environmental success probabilities (but not necessarily their actual values), it may be possible to choose λ so that the selective bootstrap algorithm, or a similar deterministic algorithm, is fast and reliable. In problems like Task

2, however, in which the success probabilities corresponding to different input patterns require different values of λ for successful learning, there is no way to ensure the complete solution of an associative reinforcement learning task even if there is some knowledge of the success probabilities. Our results show that the A_{R-P} algorithm does not suffer from these difficulties, although it is considerably slower in solving the easier tasks.

VIII. DISCUSSION

Since the associative reinforcement learning task and the A_{R-P} algorithm are hybrids of the tasks and algorithms of the theories of learning automata and pattern classification, we can discuss our result from the perspective of each theory. We first suggest how our algorithm contributes to the theory of learning automata.

A. Collective Behavior of Learning Automata

Since a learning automaton implements a global stochastic search of its action space, it is not affected by the existence of local extrema; indeed, since the action set (as usually formulated) is not topologized, the concept of locality is not even applicable. However, a major factor limiting the utility of learning automaton algorithms is that the speed of convergence decreases rapidly as the number of actions increases. Partly for this reason, theorists have become increasingly interested in the collective behavior of learning automata in which each action of the collection is a vector whose components are the actions of independent learning automata. The search in this structured space of collective actions can proceed much more rapidly than can a search by a single learning automaton having a separate action corresponding to each possible collective action. However, this structure introduces the possibility that local optima exist, and collections of learning automata are not guaranteed to converge to global optima. Nevertheless, their ability to improve performance in the absence of direct communication between component automata has suggested that such distributed "teams" of learning automata can be effective in decentralized control problems such as the routing of traffic in computer and communication systems [21].

Related to this is the behavior of learning automata as "players" in games in which success probabilities involve conflicts of interest. Here, it is usual to assume that the learning automata have no knowledge of other players' strategies or of the payoff structure of the game. Among the results in this area is that certain learning automaton algorithms converge to the von Neumann value of any zero-sum game [20], [21].

Although the ability to produce effective collective behavior in the absence of direct inter-automaton communication is an important property of learning automata, it is interesting to consider the capabilities of collections of mutually communicating learning automata. An *associative* learning automaton in a collection can receive input vectors consisting of signals produced by other automata as well as signals indicating environmental state. The latter information can be used to reduce the effect of environmental nonstationarity by allowing discrimination between environmental states, and the former information can allow more efficient search by enforcing statistical constraints on collective actions. Communication links between automata cause their actions to be statistically correlated. If appropriate communication links form, this correlation can cause trials to be concentrated in high-payoff regions of the search space, thus accelerating search while possibly avoiding convergence to a local optimum.[6] Additionally, communication among associative learning automata that act as game players may provide a basis for the formation of solutions that are cooperative in the sense of game theory [42]. All of these possibilities deserve study.

B. Learning in Networks of Pattern-Classifying Elements

Networks of elements implementing pattern-classification algorithms have been extensively studied by many researchers—motivated both by the desire to obtain more sophisticated pattern-classification abilities from networks than are obtainable from single elements, as well as by the desire to investigate formal analogs of neural networks [3], [13], [27]. One of the reasons that such research is no longer widespread is the lack of a result extending single-element learning theorems to networks. A single element implementing a

[6] This statement is suggested by possible parallels between this process and varieties of stochastic relaxation methods for constraint satisfaction, e.g., Geman and Geman [40]; Kirkpatrick *et al.* [41]; Hinton and Sejnowski [37]; Hinton *et al.* [38]; Smolensky [39].

linearly-parameterised decision rule (such as an adaline or an element implementing the perceptron algorithm) has well-known limitations regarding the class of discriminations it is able to make [43]. Although *any* decision rule can be imlemented by a suitably designed network of these elements, extension of the single-element gradient-descent learning procedure to the task of adjusting all the network's parameters fails due to the inability to determine the gradient locally and the multimodality of the error functional.[7]

Although an element implementing the A_{R-P} algorithm is also limited to forming only linear discriminations, networks of such elements behave very differently than networks of supervised learning elements. Let us distinguish a network's *output elements* from its *interior elements* (cf. refs [37] and [38]). Output elements are those whose activity is directly visible to the network's environment and that are required to assume certain values when externally-supplied input patterns are present. Interior elements are those whose activity is not directly visible and that are somehow to provide an encoding of input signals that will allow the output elements to respond correctly. The Achilles' heel of supervised learning elements as network components is that they can only learn if supplied with individualised signals indicating (perhaps probabilistically) correct responses. Although in many tasks it may be possible for the network's "teacher" to provide such information to the network's output elements (since it is these that define the network's visible response), it may not be possible for this teacher to provide such signals to the interior elements without *a priori* knowledge of the implementation details of the desired input/output function.

Since elements capable of associative reinforcement learning are able to learn under the influence of a less informative training signal than are supervised learning elements, they are able to learn effectively as interior elements of networks. If the network's environment supplies interior elements with overall assessments of the collective behavior of the network's components (based on the activity of the output elements), then the learning task

[7] Hinton and Sejnowski [37] propose a learning algorithm based on stochastic relaxation to overcome these difficulties for the case of symmetrically connected networks. This is an interesting approach, further developed by Hinton *et al.* [38], and we have not yet fully investigated the relationship between their approach to learning within networks and our own.

faced by an interior element resembles the associative reinforcement learning task. Even if the evaluation signal is a deterministic function of the activity of the output elements, it generally is not a deterministic function of the activity of individual interior elements. Moreover, this uncertainty cannot be guaranteed to be of the form that renders the task of an interior element transformable into a conventional pattern-classification task. Simulation experiments show that elements implementing algorithms similar to the A_{R-P} algorithm are able to learn effectively as interior elements of networks [5], [7], [44]. Cooperativity of the type discussed above does indeed occur. Elements form links to one another and consequently are more successful than they would be if they acted independently. Our motivation for considering the A_{R-P} algorithm is its use as a network component, and our research is continuing in this direction.[8]

IX. CONCLUSION

We have presented an algorithm that represents a synthesis of aspects of learning automaton and supervised learning pattern-classification algorithms. Since it lies in the intersection of important classes of algorithms, the A_{R-P} algorithm should help illuminate the relationship between theories that have remained largely separate. We also think that the properties combined in the algorithm complement one another in an important way. Pattern classification capabilities allow learning automata to respond to differing environmental contexts; learning automaton capabilities allow pattern-classification systems to learn under the influence of training information that is less informative than required by the conventional algorithms. This combination of capabilities should permit collections of adaptive elements to exhibit far richer forms of cooperative behavior than is possible without inter-element communication or without robust ability to search under uncertainty.

[8] We are aware of only a few earlier studies in which this approach to learning in networks was pursued. The simulations of Farley and Clarke [13] provide an isolated early example; Klopf's [10], [11], theory of the "hedonistic neuron" is falls into this class, as does the selective bootstrap adaptation algorithm of Widrow *et al.* [12]. Minsky [45] describes what amounts to a stochastic learning automaton algorithm used in a network, but it does not qualify as associative in the sense used here.

APPENDIX I: COMPARISON OF ASSOCIATIVE REINFORCEMENT LEARNING AND SUPERVISED LEARNING PATTERN CLASSIFICATION

Here we discuss some implications for algorithms of associative reinforcement learning tasks in which the success probabilities for the two actions do not sum to one for some input pattern. Consider an associative reinforcement learning task with two actions $a^{(1)} = 1$ and $a^{(2)} = -1$. Let d^{1i} denote the success probability given input pattern $x^{(i)}$ and action 1; and let d^{-1i} denote the success probability given pattern $x^{(i)}$ and action -1. Let $p^{1i} = Pr\{a_k = 1|x_k = x^{(i)}\}$. Without loss of generality for our present purpose, let us assume that $d^{1i} > d^{-1i}$, so that $a_k = 1$ is the optimal action for input pattern $x^{(i)}$. If the success/failure signal b_k has values 1 and -1, representing success and failure respectively, we can estimate a class label, or optimal action, z_k by multiplication, i.e., $z_k = b_k a_k$.

Suppose we treat this task as a supervised learning pattern-classification task whose object it is to classify each pattern x according to the predominant value of z with which that pattern is paired in a training sequence. Since (cf. (6a))

$$Pr\{z_k = 1|\theta_k, x_k = x^{(i)}\} = p^{1i}d^{1i} + (1 - p^{1i})(1 - d^{-1i}),$$

the probability that z_k has any particular value clearly depends on the action probabilities in addition to the environmentally determined success probabilities. We can see that the probability that the environment provides the optimal class label, i.e., 1, given action $a_k = 1$, is

$$Pr\{z_k = 1|x_k = x^{(i)}, a_k = 1\} = d^{1i},$$

and the probability that the teacher provides the optimal class label, given action $a_k = -1$, is

$$Pr\{z_k = 1|x_k = x^{(i)}, a_k = -1\} = 1 - d^{-1i}.$$

There are three relevant cases:

Case i: $d^{1i} > .5$ and $d^{-1i} < .5$. In this case, $1 - d^{-1i} > .5$. Thus, the probability that the environment provides the optimal class label is greater than .5 no matter which of the actions is taken. In this case, the optimal class label would also be the predominant one, and an algorithm capable of minimising the probability of misclassification in a supervised learning pattern-classification task could be expected to achieve a solution when suitably modified to compute class labels by combining success/failure information and knowledge of the actions actually taken (e.g., the selective bootstrap algorithm [12]).

Case ii: $d^{1i} > .5$ and $d^{-1i} > .5$. Here, $1 - d^{-1i} < .5$. Thus, whenever the nonoptimal action is taken, the probability that the environment will give the optimal class label is less than $.5$. Consequently, if a conventional supervised learning algorithm (suitably modified to accept success/failure information) is initially biased toward producing the incorrect action, then this bias will tend to increase, leading to convergence to the incorrect action.

Case iii: $d^{1i} < .5$ and $d^{-1i} < .5$. Here, whenever the optimal action is taken, the probability that the environment will give the optimal class label is less than $.5$; and since $1 - d^{-1i} > .5$, whenever the nonoptimal action is taken, the probability that the environment will give the optimal class label is greater than $.5$. Under these conditions, a conventional supervised learning algorithm (suitably modified to accept success/failure information) will oscillate rather than converge (cf. the behavior of the selective bootstrap algorithm in the simulated Task 2, Fig. 2b).

From this analysis, one sees that the predominant class label corresponds to the optimal one only if for each $x \in X$, it is either true that $d(x, a^{(1)}) > .5$ and $d(x, a^{(2)}) < .5$, or $d(x, a^{(1)}) < .5$ and $d(x, a^{(2)}) > .5$. This is certainly true when $d(x, a^{(1)}) + d(x, a^{(2)}) = 1$ for all $x \in X$, but is not be true in a general associative reinforcement learning task. Consequently, a conventional algorithm for supervised learning pattern-classification, modified to accept success/failure information rather than class labels, can fail completely in these more subtle reinforcement learning tasks. The difficulties these algorithms encounter in cases ii and iii parallel the difficulties encountered by nonassociative learning automaton alogithms that are deterministic or are fixed-structure stochastic (ref. [36] provides good illustrations of these difficulties).

APPENDIX II: CONVERGENCE PROOF

Here we prove the convergence theorem stated in Section V. The proof is an extension of the strong convergence proof due to Lakshmivarahan (Theorem 4.1 in ref. [1], p. 112; also extended are the proofs of supporting results in ref. [1], pp. 24, 26, and ref. [2]). Another proof provided by Lakshmivarahan [1] which can also be extended to the associative case is based on Kushner's method of asymptotic analysis as described in ref. [46]. We first state some properties of the algorithm.

(P1): Letting

$$\begin{aligned} p^{1i}(\theta) &= Pr\{a_k = 1|\theta_k = \theta, x_k = x^{(i)}\} \\ &= 1 - \Psi(-\theta^{\mathrm{T}} x^{(i)}), \end{aligned}$$

we have that

$$E\{a_k|\theta_k = \theta, x_k = x^{(i)}\} = 2p^{1i}(\theta) - 1.$$

Note that since Ψ is continuous and strictly monotonic, so is p^{1i} regarded as a function of $\theta^{\mathrm{T}} x^{(i)}$.

(P2): Let $\delta\theta_k = \theta_{k+1} - \theta_k$. Then

$$E\{\delta\theta_k|\theta_k = \theta\} = \rho_k w(\theta, \lambda),$$

where

$$\begin{aligned} w(\theta, \lambda) &= \sum_{i=1}^{m} \xi^i w^i(\theta, \lambda) x^{(i)} \\ w^i(\theta, \lambda) &= w^{Ri}(\theta) + \lambda w^{Pi}(\theta), \end{aligned}$$

and

$$\begin{aligned} 2\rho_k w^{Ri}(\theta) x^{(i)} &= E\{\delta\theta_k|\theta_k = \theta, x_k = x^{(i)}, b_k = 1\} \\ 2\lambda\rho_k w^{Pi}(\theta) x^{(i)} &= E\{\delta\theta_k|\theta_k = \theta, x_k = x^{(i)}, b_k = -1\}. \end{aligned}$$

It can be easily shown that

$$\begin{aligned} w^{Ri}(\theta) &= p^{1i}(\theta)(1 - p^{1i}(\theta))(d^{1i} - d^{-1i}) \\ w^{Pi}(\theta) &= (1 - p^{1i}(\theta))^2 c^{-1i} - (p^{1i}(\theta))^2 c^{1i}, \end{aligned}$$

where

$$d^{1i} = d(x^{(i)}, 1) = 1 - c^{1i}$$
$$d^{-1i} = d(x^{(i)}, -1) = 1 - c^{-1i}.$$

(P3): If the initial state of the A_{R-P} algorithm is θ_1, then for $k > 1$,

$$\theta_k = \theta_1 + \sum_{j=1}^{k-1} \delta\theta_j.$$

Since each $\delta\theta_j$ is a scalar times x_j, $\theta_k - \theta_1$ is in the linear space spanned by the set X of input vectors; that is, letting M denote the matrix whose columns are the input patterns and letting $R(M)$ denote the range of M, $\theta_k \in R(M) + \theta_1$, for $k > 1$.

Lemma 1. *For $\lambda = 1$ and for each i, $1 \le i \le m$, there exist unique $\hat{\beta}^i$ and $\beta^i \in (0,1)$ such that, for any θ, if $p^{1i}(\theta) = \hat{\beta}^i$, then $w^{Pi}(\theta) = 0$; and if $p^{1i}(\theta) = \beta^i$, then $w^i(\theta, 1) = 0$; where*

$$\text{if} \quad d^{1i} > d^{-1i}, \quad \text{then} \quad \beta^i > \hat{\beta}^i > .5;$$
$$\text{if} \quad d^{1i} < d^{-1i}, \quad \text{then} \quad \beta^i < \hat{\beta}^i < .5.$$

Proof: We prove this for the case $d^{1i} > d^{-1i}$; the proof for the other case is similar. From (P2) we have that

$$\text{if} \quad p^{1i}(\theta) = 0, \quad \text{then} \quad w^{Pi}(\theta) = w^i(\theta, 1) = c^{-1i} > 0;$$
$$\text{if} \quad p^{1i}(\theta) = 1, \quad \text{then} \quad w^{Pi}(\theta) = w^i(\theta, 1) = -c^{1i} < 0;$$
$$\text{if} \quad p^{1i}(\theta) = .5, \quad \text{then} \quad w^{Pi}(\theta) = 1/4(c^{-1i} - c^{1i}) > 0.$$

Since w^{Pi} is a quadratic function of $p^{1i}(\theta)$ and $p^{1i}(\theta) \in [0,1]$ for all θ, the above facts imply that there exists a unique $\hat{\beta}^i \in (.5, 1]$ such that when $p^{1i}(\theta) = \hat{\beta}^i$, $w^{Pi}(\theta) = 0$. Further, note that for $p^{1i}(\theta) \in (0,1)$, $w^{Ri}(\theta) > 0$. Hence, if $p^{1i}(\theta) = \hat{\beta}^i$, then

$$w^i(\theta, 1) = w^{Ri}(\theta) + w^{Pi}(\theta) > w^{Pi}(\theta) > 0;$$

and there exists a unique $\beta^i \in (\hat{\beta}^i, 1)$ such that when $p^{1i}(\theta) = \beta^i$, $w^i(\theta, 1) = 0$.

For future reference, also observe that

$$\begin{aligned} &\text{if} \quad p^{1i}(\theta) > \hat{\beta}^i, \quad \text{then} \quad w^{Pi}(\theta) > 0; \\ &\text{if} \quad p^{1i}(\theta) < \hat{\beta}^i, \quad \text{then} \quad w^{Pi}(\theta) < 0. \end{aligned} \tag{11}$$

Q.E.D.

Lemma 2. *For each* $\lambda \in (0,1)$ and *for each* i, $1 \le i \le m$, *there exists* a *unique* β^i_λ *such that*

$$\begin{aligned} &\text{if} \quad p^{1i}(\theta) > \beta^i_\lambda, \quad \text{then} \quad w^i(\theta,\lambda) < 0; \\ &\text{if} \quad p^{1i}(\theta) = \beta^i_\lambda, \quad \text{then} \quad w^i(\theta,\lambda) = 0; \\ &\text{if} \quad p^{1i}(\theta) < \beta^i_\lambda, \quad \text{then} \quad w^i(\theta,\lambda) > 0. \end{aligned} \tag{12}$$

Further, if $d^{1i} > d^{-1i}$, *then*

$$\beta^i_\lambda > \beta^i \quad \text{and} \quad \lim_{\lambda \to 0} \beta^i_\lambda = 1;$$

whereas if $d^{1i} < d^{-1i}$, *then*

$$\beta^i_\lambda < \beta^i \quad \text{and} \quad \lim_{\lambda \to 0} \beta^i_\lambda = 0.$$

Proof: Again we prove this for the case $d^{1i} > d^{-1i}$; the proof for the other case is similar. By (P2) we have for each i, $1 \le i \le m$, that

$$\begin{aligned} w^i(\theta,\lambda) &= w^{Ri}(\theta) + \lambda w^{Pi}(\theta) \\ &= w^{Ri}(\theta) + w^{Pi}(\theta) - (1-\lambda) w^{Pi}(\theta) \\ &= w^i(\theta,1) - (1-\lambda) w^{Pi}(\theta). \end{aligned}$$

Hence, from the definition of β^i and (11) in Lemma 1, we have that if $p^{1i}(\theta) = \beta^i$, then

$$w^i(\theta,\lambda) = -(1-\lambda) w^{Pi}(\theta) > 0;$$

and that if $p^{1i}(\theta) = 1$, then

$$w^i(\theta,\lambda) = -\lambda c^{1i} < 0.$$

Thus the effect of multiplying $w^{Pi}(\theta)$ by λ is to shift the zero-crossing β^i further to the right. Consequently, there exists a unique $\beta^i_\lambda > \beta^i$ such that if $p^{1i}(\theta) = \beta^i_\lambda$, then $w^i(\theta,\lambda) = 0$. It is clear that the other parts of (12) hold as well.

Further, for any $\lambda' < \lambda$, it can be seen that if $p^{1i}(\theta) = \beta^i_\lambda$, then

$$w^i(\theta, \lambda') = -(\lambda - \lambda') w^{P^i}(\theta) > 0.$$

This implies that β^i_λ increases as λ decreases. Since $w^i(\theta, 0) = 0$ when $p^{1i}(\theta) = 1$, the least upper bound of β^i_λ is 1. Hence, $\lim_{\lambda \to 0} \beta^i_\lambda = 1$.

Q.E.D.

Lemma 3. *Under conditions (C1)-(C3), for each $\lambda \in (0, 1]$ there exists a unique $\theta^\circ_\lambda \in R(M) + \theta_1$ such that $w(\theta^\circ_\lambda, \lambda) = 0$; and for all i, $1 \le i \le m$,*

$$\lim_{\lambda \to 0} p^{1i}(\theta^\circ_\lambda) = \begin{cases} 1, & \text{if } d^{1i} > d^{-1i}; \\ 0, & \text{if } d^{1i} < d^{-1i}. \end{cases}$$

Proof: Since X is a linearly independent set and $\xi^i > 0$, $1 \le i \le m$,

$$w(\theta, \lambda) = \sum_{i=1}^{m} \xi^i w^i(\theta, \lambda) x^{(i)} = 0 \iff w^i(\theta, \lambda) = 0, \quad 1 \le i \le m.$$

From Lemma 2 we know that for all i, $1 \le i \le m$, there exists a unique β^i_λ such that

$$p^{1i}(\theta) = \beta^i_\lambda \Longrightarrow w^i(\theta, \lambda) = 0. \tag{13}$$

Since p^{1i} is strictly monotonic and continuous as a function of $\theta^{\mathrm{T}} x^{(i)}$, it is invertible so that (13) implies that for all i, $1 \le i \le m$, there exists a unique $\bar{\beta}^i_\lambda$ such that $p^{1i}(\theta) = \beta^i_\lambda$ when $\theta^{\mathrm{T}} x^{(i)} = \bar{\beta}^i_\lambda$. Letting $\bar{\beta}_\lambda$ denote the vector whose components are $\bar{\beta}^i_\lambda$, this can be written

$$M^{\mathrm{T}} \theta = \bar{\beta}_\lambda, \tag{14}$$

where M is the matrix whose columns are the input patterns. Since the rows of M^{T} are independent, there is at least one θ°_λ that satisfies (14). Additionally, the condition that $\theta^\circ_\lambda \in R(M) + \theta_1$ implies that (14) can be written

$$M^{\mathrm{T}}(M\gamma + \theta_1) = \bar{\beta}_\lambda,$$

for some vector $\gamma \in \Re^m$. Since the columns of M are independent, $M^T M$ is invertible and γ is the unique vector

$$\gamma = (M^T M)^{-1}(\tilde{\beta}_\lambda - M^T \theta_1).$$

Thus, $\theta_\lambda^\circ = M\gamma + \theta_1$ is the unique vector that satisfies (14), and thus is the unique vector in $R(M) + \theta_1$ for which $w(\theta_\lambda^\circ, \lambda) = 0$.

Finally, combining (13) with Lemma 2, we have that and for all i, $1 \leq i \leq m$,

$$\lim_{\lambda \to 0} p^{1i}(\theta_\lambda^\circ) = \beta_\lambda^i = \begin{cases} 1, & \text{if } d^{1i} > d^{-1i}; \\ 0, & \text{if } d^{1i} < d^{-1i}. \end{cases}$$

Q.E.D.

Lemma 4. *Under conditions (C1)–(C3),* $(\theta - \theta_\lambda^\circ)^T w(\theta, \lambda) \leq 0$ *for all* $\lambda \in (0, 1]$ *and for all* $\theta \in \Re^n$. *Further,* θ_λ° *is the unique* θ *in* $R(M) + \theta_1$ *such that* $(\theta - \theta_\lambda^\circ)^T w(\theta, \lambda) = 0$.

Proof: From (12) in Lemma 2 and the monotonicity of $p^{1i}(\theta)$, we have for all $\lambda \in (0, 1]$ that

$$\begin{aligned} w^i(\theta, \lambda) < 0 &\Longleftrightarrow \theta^T x^{(i)} > \theta_\lambda^{\circ T} x^{(i)}; \\ w^i(\theta, \lambda) = 0 &\Longleftrightarrow \theta^T x^{(i)} = \theta_\lambda^{\circ T} x^{(i)}; \\ \text{and} \quad w^i(\theta, \lambda) > 0 &\Longleftrightarrow \theta^T x^{(i)} < \theta_\lambda^{\circ T} x^{(i)}. \end{aligned} \tag{15}$$

Additionally,

$$\begin{aligned} (\theta - \theta_\lambda^\circ)^T w(\theta, \lambda) &= (\theta - \theta_\lambda^\circ)^T \sum_{i=1}^m \xi^i w^i(\theta, \lambda) x^{(i)} \\ &= \sum_{i=1}^m \xi^i w^i(\theta, \lambda)[\theta^T x^{(i)} - \theta_\lambda^{\circ T} x^{(i)}]. \end{aligned} \tag{16}$$

Combining (15) and (16), we have that

$$(\theta - \theta_\lambda^\circ)^T w(\theta, \lambda) \leq 0, \quad \theta \in \Re^n. \tag{17}$$

From Lemma 3 we have that for any $\lambda \in (0, 1]$, θ_λ° is the unique vector in $R(M) + \theta_1$ such that $w^i(\theta_\lambda^\circ, \lambda) = 0$ for all i, $1 \leq i \leq m$. Hence θ_λ° is the unique vector in $R(M) + \theta_1$ such that equality in (17) holds.

Q.E.D.

We now derive an additional property of the algorithm. For each i, $1 \leq i \leq m$, let

$$\begin{aligned}
a^{Ri}(\theta) &= p^{1i}(\theta)(1-p^{1i}(\theta))[(1-p^{1i}(\theta))d^{1i} + p^{1i}(\theta)d^{-1i}]; \\
a^{Pi}(\theta) &= (p^{1i}(\theta))^3 c^{1i} + (1-p^{1i}(\theta))^3 c^{-1i}; \\
a^i(\theta) &= a^{Ri}(\theta) + \lambda^2 a^{Pi}(\theta); \\
\text{and} \quad a(\theta) &= \sum_{i=1}^{m} \xi^i a^i(\theta) \|x^{(i)}\|^2.
\end{aligned}$$

Then

$$\begin{aligned}
&E\{\delta\theta_k^T \delta\theta_k | \theta_k = \theta\} = \rho_k^2 a(\theta); \\
&E\{\delta\theta_k^T \delta\theta_k | \theta_k = \theta, x_k = x^{(i)}, b_k = 1\} = \rho_k^2 \|x^{(i)}\|^2 a^{Ri}(\theta); \\
\text{and} \quad &E\{\delta\theta_k^T \delta\theta_k | \theta_k = \theta, x_k = x^{(i)}, b_k = -1\} = \lambda^2 \rho_k^2 \|x^{(i)}\|^2 a^{Pi}(\theta).
\end{aligned}$$

Since $p^{1i}(\theta) \in [0,1]$ for all $x^{(i)}$ and θ, for any given set of $x^{(i)}$, there exist constants L_1 and L_2 such that

$$a(\theta) < L_1 \sum_{i=1}^{m} \|x^{(i)}\|^2 < L_2.$$

Hence,

$$E\{\delta\theta_k^T \delta\theta_k | \theta_k = \theta\} < L_2 \rho_k^2. \tag{18}$$

We are now able to apply a version of the convergence theorem stated in Section III for stochastic approximation to prove that under the conditions (C1)-(C4), θ_k converges to θ_λ° with probability 1. A proof, omitted here, can be found in refs. [1] or [27]. Here we establish that the conditons for the theorem hold.

Lemma 5. *Under conditions (C1)-(C4), the random process $\{\theta_k\}_{k\geq 1}$ generated by the A_{R-P} algorithm converges to θ_λ° with probability 1.*

Proof: We show that the random process $\{\theta_k\}_{k\geq 1}$ satisfies the conditons (A1)-(A4) presented in the discussion of Section III (after [27]) of supervised learning pattern classification. (The roles of $y(\theta)$ and $r(\theta)$ of Section III are here played respectively by $E\{\delta\theta_k | \theta_k = \theta\}$ and $w(\theta, \lambda)$.)

First note that by (P3), $\theta_k \in R(M) + \theta_1$, $k > 1$. By Lemma 3 we have for $\theta \in R(M) + \theta_1$ and $\lambda \in (0,1]$, that $w(\theta,\lambda) = \frac{1}{\rho_k} E\{\delta\theta_k | \theta_k = \theta\}$, $k > 1$, has a unique sero at θ_λ°. This is as required by property (A1) above.

Condition (A2) is just conditon (C4).

By Lemma 4 we have that $(\theta - \theta^{\circ}_{\lambda})^{T} w(\theta, \lambda) < 0$, for all $\theta \in R(M) + \theta_1$ such that $\theta \neq \theta^{\circ}_{\lambda}$. Given the definition of $\delta\theta$, this implies (A3) above.

By (18), $E\{\|\delta\theta\|^2|\theta\} < L_2\rho^2_k$. This is a stronger condition than (A4) above. It is the analog of condition 4.8 of Lakshmivarahan (ref. [1], p. 112).

The remainder of the proof follows the argument given by Kasyap *et al.* [4] based on that of Gladyshev [47], or Theorem 4.1 of Lakshmivarahan (ref. [1], p. 112).

Q.E.D.

Our convergence theorem follows immediately from Lemmas 1, 3 and 5.

REFERENCES

[1] S. Lakshmivarahan, *Learning Algorithms and Applications.* New York: Springer-Verlag, 1981.

[2] S. Lakshmivarahan, "ϵ-optimal learning algorithms—Non-absorbing barrier type," Technical Report EECS 7901, School of Electrical Engineering and Computer Sciences, University of Oklahoma, Norman Oklahoma, 1979.

[3] F. Rosenblatt, *Principles of Neurodynamics.* New York: Spartan Books, 1962.

[4] R. L. Kasyap, C. C. Blaydon, and K. S. Fu, "Stochastic approximation," in *Adaptation, Learning. and Pattern Recognition Systems: Theory and Applications*, J. M. Mendel and K. S. Fu, Eds. New York: Academic Press, 1970.

[5] A. G. Barto, Editor. "Simulation experiments with goal-seeking adaptive elements," Air Force Wright Aeronautical Laboratories/Avionics Laboratory Technical Report AFWAL-TR-84-1022, Wright-Patterson AFB, Ohio, 1984.

[6] A. G. Barto, C. W. Anderson, and R. S. Sutton, "Synthesis of nonlinear control surfaces by a layered associative search network," *Biol. Cybern.*, vol. 43, pp. 175–185, 1982.

[7] A. G. Barto and R. S. Sutton, "Landmark learning: An illustration of associative search," *Biol. Cybern.*, vol. 42, pp. 1–8, 1981.

[8] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike elements that can solve difficult learning control problems," *IEEE Trans. on Syst., Man, Cybern.*, vol. SMC-13, pp. 834–846, 1983.

[9] A. G. Barto, R. S. Sutton, and P. S. Brouwer, "Associative search network: A reinforcement learning associative memory," *Biol. Cybern.*, vol. 40, pp 201–211, 1981.

[10] A. H. Klopf, "Brain function and adaptive systems- A heterostatic theory," Air

Force Cambridge Res. Lab. Res. Rep., AFCRL-72-0164, Bedford, MA., 1972 (A summary appears in *Proc. Int. Conf. on Syst., Man, Cybern.*, 1974).

[11] A. H. Klopf, *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence.* Washington, D.C.: Hemisphere, 1982.

[12] B. Widrow, N. K. Gupta, and S. Maitra, "Punish/reward: Learning with a critic in adaptive threshold systems," *IEEE Trans. on Syst., Man, Cybern.*, vol. 5, pp. 455–465, 1973.

[13] B. G. Farley and W. A. Clark, "Simulation of self-organising systems by digital computer," *I.R.E. Trans. Inf. Theory*, vol. 4, pp. 76–84, 1954.

[14] W. K. Estes, "Toward a statistical theory of learning," *Psychol. Rev.*, vol. 57, pp. 94–107, 1950.

[15] R. R. Bush and F. Mosteller, *Stochastic Models for Learning.* New York: Wiley, 1955.

[16] R. R. Bush and W. K. Estes, Eds., *Studies in Mathematical Learning Theory.* Stanford: Stanford University Press, 1959.

[17] R. C. Atkinson and W. K. Estes, "Stimulus sampling theory," in *Handbook of mathematical psychology, Vol. II*, R.D. Luce, R.R. Bush, and E. Galanter, Eds. New York: Wiley, 1963.

[18] R. A. Jarvis, "Adaptive global search in a time varying environment using probabilistic automaton with pattern recognition supervision," *IEEE Trans. Syst., Sci., Cybern.*, vol. 6, pp. 209–217, 1970.

[19] M. L. Tsetlin, *Automaton Theory and Modelling of Biological Systems.* New York: Academic Press, 1973.

[20] K. S. Narendra and M. A. L. Thathachar, "Learning automata—a survey," *IEEE Trans. Syst., Man, Cybern.*, vol. 4, pp. 323–334, 1974.

[21] K. S. Narendra and S. Lakshmivarahan, "Learning automata—a critique," *J. Cybern. and Inf. Sci.,* vol. 1, pp. 53–65, 1977.

[22] H. Robbins, "A sequential decision problem with finite memory," *Proc. Nat. Acad. Sci.,* vol. 30, pp. 920–923, 1956.

[23] T. M. Cover and M. E. Hellman, "The two-armed bandit problem with time-invariant finite memory," *IEEE Trans. Inf. Theory,* vol. 16, pp. 185–195, 1970.

[24] I. H. Witten, "An adaptive optimal controller for discrete-time markov environments," *Inf. and Contr.*, vol. 34, pp. 286-295, 1977.

[25] R. A. Jarvis, "Teaching a stochastic automaton to skillfully play hand/eye games," *J. of Cybern. and Inf. Sci.*, vol. 1, pp. 161–177, 1977.

[26] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[27] G. Hinton and J. Anderson, *Parallel Models of Associative Memory.* Hilsdale, N. J.: Erlbaum, 1981.

[28] P. R. Cohen and E. A. Feigenbaum, *The Handbook of Artificial Intelligence, Vol. 3.* Los Altos, California: Kauffman, 1982.

[29] A. Albert and L. Gardner, *Stochastic Approximation and Nonlinear Regression.* Cambridge, MA: M.I.T. Press, 1967.

[30] B. Widrow and M. E. Hoff, "Adaptive switching circuits," *1960 WESCON Convention Record Part IV,* pp. 96–104, 1960.

[31] C. Derman, *Finite State Markovian Decision Processes.* New York: Academic Press, 1970.

[32] H. Mine and S. Osaki, *Markovian Decision Processes.* New York: American Elsevier Publishing Company, Inc., 1970.

[33] R. S. Sutton, "Temporal aspects of credit assignment in reinforcement learning,"

Univ. of Massachusetts Ph.D. Dissertation, 1984.

[34] M. F. Norman, "Markovian learning processes," *SIAM Review*, vol. 16, pp. 143–162, 1974.

[35] M. F. Norman, "A central limit theorem for Markov processes that move by small steps," *The Annals of Prob.*, vol. 2, pp. 1065–1074, 1974.

[36] P. Marrs and E. J. Poppelbaum, *Stochastic and Deterministic Averaging Processors*. London: The Institute of Electrical Engineers, 1981.

[37] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, "Boltzmann machines: Constraint satisfaction networks that learn," Technical Report CMU-CS-84-119, Carnegie-Mellon University, Pittsburgh, PA, 1984.

[38] G. E. Hinton and T. J. Sejnowski, "Analyzing cooperative computation," *Proc. Fifth Ann. Conf. of the Cognitive Science Society*, Rochester N.Y., 1983.

[39] P. Smolenski, "Schema selection and stochastic inference in modular environments," *Proc. Nat. Conf. on Artificial Intelligence AAAI-83*, Washington, DC, pp. 109–113, 1983.

[40] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEEE Trans. PAMI* (in press).

[41] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[42] R. D. Luce and H. Raiffa, *Games and Decisions*. New York: Wiley 1957.

[43] M. L. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969.

[44] C. W. Anderson, "Feature generation and selection by a layered network of reinforcement learning elements: Some initial experiments," COINS Technical Report 82-12, University of Massachusetts, Amherst, 1982.

[45] M. L. Minsky, "Theory of neural-analog reinforcement systems and its application to the brain-model problem," Princeton Univ. Ph.D. Dissertation. 1954.

[46] H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, New York: Springer-Verlag, 1978.

[47] E. A. Gladyshev, "On stochastic approximation," *Theory of Prob. and Appl.*, vol 10, 1965.