Associative Reinforcement Learning of Real-valued Functions

Vijaykumar Gullapalli
Computer and Information Science Department
University of Massachusetts

COINS Technical Report 90-129
May 7, 1990

Associative Reinforcement Learning of Real-valued Functions

Vijaykumar Gullapalli

May 7, 1990

Abstract

Associative reinforcement learning tasks defined by Barto and Anandan [4] combine elements of problems involving optimization under uncertainty, studied by learning automata theorists, and supervised learning pattern-classification. In our previous work, we presented the SRV algorithm [15] which had been designed for extended versions of associative reinforcement learning tasks wherein the learning system's outputs could take on real values. In this paper, we state and prove a strong convergence theorem that implies a form of optimal performance (under certain conditions) of the SRV algorithm on these tasks. Simulation results are presented to illustrate the convergence behavior of the algorithm under the conditions of the theorem. The robustness of the

This material is based upon work supported by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526 and by the National Science Foundation under Grant ECS-8912623.

algorithm is also demonstrated by simulations in which some of the conditions of the theorem are violated.

1. Introduction

Associative reinforcement learning tasks defined by Barto and Anandan [4] combine elements of problems involving optimization under uncertainty, studied by learning automata theorists, and supervised learning pattern-classification. Each of these component classes of tasks is of interest in its own right and has been studied extensively over the past three decades [20, 10]. In our previous work, we described a class of learning tasks that are extensions of the associative reinforcement learning tasks defined by Barto and Anandan, and an algorithm, called the *Stochastic Real-Valued unit* (SRV) algorithm [14, 15], that had been designed specifically for such tasks. In this paper we state and prove a strong convergence theorem that implies a form of optimal performance for a modified version of the SRV algorithm on these tasks. The proof of the theorem is based on Martingale theory and is modeled after Gladyshev's [13] proof of convergence of the Robbins-Monro process [21]. Modifications to the original algorithm were necessary to make rigorous analysis possible. However, we informally argue that the original algorithm has convergence properties similar to those of the modified algorithm, and we present simulation results which support this claim.

In associative reinforcement learning tasks, the learning system interacts in a closed loop with its environment. At each time step, the environment provides the learning system with input $\mathbf{x}$ chosen from a set of inputs, X. Using this input, the learning system selects an output z from a set of permissible outputs, Z. Based on both $\mathbf{x}$ and z, the environment computes and returns a payoff value or "reinforcement", $r \in R$. Ideally, we would like the system to learn to respond to each input with the output that has the highest expected payoff. In keeping with the earlier work on learning automata, Barto and Anandan defined associative reinforcement learning tasks as involving selection of one of a finite set of outputs (Z is a finite), and for which the payoff is a binary-valued success/failure signal (i.e. $R = \{0, 1\}$). Their interest in such tasks led to the development of the A_{R-P} (associative reward-penalty) algorithm, which they prove has a form of optimal performance in associative reinforcement learning tasks. The above conditions on the output and the payoff, however, are rather restrictive for most applications and one would like to extend the definition of associative reinforcement learning tasks to permit continuous-valued outputs and evaluations. Barto and Jordan [5] present a version of the A_{R-P} algorithm that can handle the latter case, where the evaluation returned by the environment can take on bounded continuous values (the so-called S-model case [8]). But learning continuous outputs is more difficult and existing algorithms cannot be easily extended to do so. For example, A_{R-P} units (as defined in [4]) compute their binary-valued outputs by adding noise to their activations and thresholding the sum. Such units could be easily modified to produce continuous outputs by omitting the thresholding. Unfortunately, in such units there would be no control over the amount of noise that is added to the activation and hence they would continue to produce random output values regardless of the duration of training. The search for a suitable algorithm for associative reinforcement learning tasks with continuous outputs and evaluations led to the development of the SRV algorithm [14, 15], an algorithm closely related to standard stochastic approximation procedures like the Robbins-Monro procedure [21] and the Kiefer-Wolfowitz procedure [27] and their generalizations [11]. In this algorithm, the learning system computes its real-valued output as some function of a random activation generated using the

Gaussian distribution. The activation at any time depends on the two parameters, the mean and the standard deviation, used in the Gaussian distribution, which, in turn, depend on the current inputs to the unit. The SRV algorithm adjusts these two parameters so as to increase the probability of producing the optimal real value for each input pattern. The algorithm does this by maintaining the mean of its activation as an estimate of the *optimal* activation[1] and using the standard deviation to control the amount of search around the current mean value of the activation.

Several algorithms related to the SRV algorithm have been described in the literature, although, to the best of our knowledge, no strong convergence results have been proven for any of them. Williams [25], [26] describes a very similar algorithm involving the use of the Gaussian distribution to generate real-valued outputs. His approach, unlike ours, is based on using the derivatives of the logarithm of the Gaussian distribution function to update the mean and standard deviation. Williams shows that, on an average, using these derivatives to update the parameters results in an increase in the expected evaluation from the environment. Our algorithm also has components analogous to an earlier algorithm of Sutton [23], which also uses the Gaussian distribution to generate real-valued outputs. However, in his algorithm, the output is controlled by varying the mean alone and the standard deviation is held constant. Harth and Tzanakou [17], in a somewhat equivalent approach, designed an algorithm, called Alopex, to produce real values that are proportional to a bias with random noise added to it. In the original algorithm, the noise had a fixed distribution, although in

[1]Informally, the activation that has the maximum expectation of reinforcement from the environment. A formal definition of the optimal action is given in Section 2.

more recent versions feedback is used to adjust the noise distribution. Farley and Clark [12] also describe a scheme in which the noise level is varied based on changes in performance over previous time steps. More recently, Alspector, Allen, Hu, and Satyanarayana [2] described a reinforcement learning algorithm based on the Boltzmann machine learning algorithm [1], in which the noise in the activation of their stochastic units is controlled so as to keep the units active for a reasonable fraction of the time. In addition to the frequency of activation of the units, they also used measures such as the sum of the magnitudes of weights into a unit and the past history of reinforcement received by the network to determine the amplitude of the noise.

Before presenting the algorithm and the associated convergence result, we briefly describe the relevant aspects of stochastic learning automata theory, work in pattern classification, and stochastic approximation theory. This description provides a background for the approach we have taken in designing the SRV algorithm. We also present, in more formal terms, the extended definition used in this paper of associative reinforcement learning tasks.

2. Associative reinforcement learning

Associative reinforcement learning tasks, as we define here, involve the following interaction between the environment and the learning system. At time step t the environment provides the learning system with some context vector $\mathbf{x}(t)$ selected from a set of vectors $X \subseteq \Re^n$, where $\Re$ is the set of real numbers. Based on this input, the learning system produces a random output $z(t)$ selected according to some internal probability distribution over some interval $Z \subseteq \Re$. The environment evaluates the output $z(t)$ in the context of the input $\mathbf{x}(t)$ and sends to the learning system an evaluation signal $r(t) \in R = [0,1]$, with $r(t) = 1$ denoting the maximum evaluation. This evaluation is determined according to some conditional probability distribution $G : R \times X \times Z \rightarrow [0,1]$, where $G(r, \mathbf{x}, z) = Pr\{r(t) \leq r \mid \mathbf{x}(t) = \mathbf{x}, z(t) = z\}$. The objective of the learning system is to learn to respond to each input pattern $\mathbf{x} \in X$ with the action $z^{\mathbf{x}} \in Z$ with probability 1, where $z^{\mathbf{x}}$ is such that $E(r \mid \mathbf{x}, z^{\mathbf{x}}) = \max_{z \in Z}\{E(r \mid \mathbf{x}, z)\}$.

Barto and Anandan [4] had defined associative reinforcement learning tasks for the case when the output set Z is finite and the environmental evaluation r is binary-valued. The definition of associative reinforcement learning tasks given above is a direct generalization of their definition to the case when the output z can take on continuous values and the environmental evaluation r lies in the interval $[0,1]$. It is also possible to reduce associative reinforcement learning tasks to other classes of commonly studied tasks by placing restrictions on various aspects of the task definition. For example, in the case of a single context vector ($\mid X \mid = 1$), these tasks become members of an extensively studied class called learning automata tasks. Simple learning automata operating in stochastic environments have gained attention as models of learning since the work of Tsetlin [24] and that of psychologists studying mathematical learning theory (e.g., [7] and [3]). A good review of the theory of stochastic learning automata is provided by Narendra and Thathachar [20]. A stochastic learning automaton interacts with its environment by randomly selecting an action as output to the environment, which, in turn, produces a random evaluation of the action. This evaluation (also known as a "success signal", "payoff", "reward", or "reinforcement") is used by the automaton to update its action probabilities. Learning involves adjusting the action probabilities so as to increase the expectation of favorable evaluations for future actions. Learning automata can be further classified according to the kind of evaluation they receive from the environment. If the set of possible evaluations is binary (denoting success/failure), the automaton is called a P-model automaton. The evaluation is drawn from a finite set of more than two values for a Q-model automaton and from an interval of the real line (usually [0,1]) for an S-model automaton. The environment is said to be *stationary* if the probability distribution used to produce the evaluation is constant over time. Otherwise, it is said to be *nonstationary*.

For the purposes of this paper, it is important to note that the only input to the learning automaton, as defined above, is the evaluation signal. As long as the environment is stationary, such an automaton can learn to select the optimal action using just the evaluation. But if we consider the task of learning optimal actions in different situations using such an automaton, it is clear that the automaton needs to consider input other than the evaluation signal. Such *context input* [6] serves to indicate the current state of the environment to the automaton. Since the probability distribution used to produce the evaluation may depend on the state of the environment, the automaton has to use the context input to select the preferred action appropriate for that context. In other words, the automaton has to learn an associative map from the context input to the preferred action for that context.

Associative reinforcement learning tasks can also be reduced to a special class of tasks involving learning associative maps, called supervised learning pattern classification tasks.

This can be done by restricting both the output and the evaluation to discrete values and defining the evaluation to be a deterministic function of the input alone. Specifically, the "evaluation" has to be the label of the class of vectors or patterns in the input space to which the input belongs. Pattern classification systems have been the subject of intense study since the 1950s. A comprehensive overview of pattern classification techniques is provided by Duda and Hart [10]. Learning in a pattern classification system involves using pattern – class-label pairs provided as training input to the system to develop a classification rule that assigns the correct class label for each pattern or, in general, minimizes the probability of misclassification. It is usually assumed that the training pairs are produced randomly by the environment in which the classification system operates. Formally, for an m-class problem, the class ω_i is assumed to occur with probability $P(\omega_i)$, $1 \leq i \leq m$, and a particular pattern vector $\mathbf{x}$ from the i^{th} class is assumed to occur with probability density $p(\mathbf{x} \mid \omega_i)$. Therefore the pattern classification problem is reduced to computing the *a posteriori* probability

$$P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{m} p(\mathbf{x} \mid \omega_j)P(\omega_j)} \tag{1}$$

for every i and selecting as the class label that ω_i for which $P(\omega_i \mid \mathbf{x})$ is maximum. Equivalently, one can compute the *discriminant functions*

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid \omega_i)P(\omega_i) \tag{2}$$

and select the label of the class with the largest discriminant function value.

In one set of techniques for pattern classification, called *linear discriminant function* techniques [10], the discriminant functions are linear functions of the input pattern of the form

$$g_i(\mathbf{x}) = \boldsymbol{\theta}_i^{\mathsf{T}}\mathbf{x} + c_i, \tag{3}$$

where θ_i, $1 \leq i \leq m$, are weight vectors. For the two class case, this is equivalent to forming a single discriminant function

$$g(\mathbf{x}) = \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} + c, \tag{4}$$

so that $\mathbf{x}$ is assigned to ω_1 if $g(\mathbf{x})$ is > 0 and to ω_2 otherwise. Solving the classification problem now involves using the training data to find weights that minimize the probability of misclassification. Given randomly generated training pairs, if we can define a suitable error functional that quantifies the classification error, we can apply known stochastic approximation methods [18] to this problem. These methods have been developed for finding a set of parameters $\boldsymbol{\theta}$ that minimize (or maximize) a criterion function $J(\boldsymbol{\theta})$ in situations where observations of the function values for any given setting of parameters are corrupted with noise. It is assumed that we can observe either the random variable $z(\boldsymbol{\theta})$ which is such that $E[z(\boldsymbol{\theta}) \mid \boldsymbol{\theta}] = J(\boldsymbol{\theta})$ or the random variable $y(\boldsymbol{\theta})$ which satisfies the relation $E[y(\boldsymbol{\theta}) \mid \boldsymbol{\theta}] = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ (i.e. $y(\boldsymbol{\theta})$ is a noisy measurement of the gradient of the criterion function). When the gradient information $y(\boldsymbol{\theta})$ is available, the Robbins-Monro algorithm [21] or its generalizations can be applied to update the parameters $\boldsymbol{\theta}$ so that the criterion function is optimized. Otherwise, the Kiefer-Wolfowitz [19] algorithm, or other similar algorithms based on obtaining estimates of the gradient from the noisy observations of the function values, $z(\boldsymbol{\theta})$, can be applied.

Our efforts have been directed towards incorporating the relevant techniques from the areas described above in a single learning system that (1) learns the action that yields the highest evaluation in a given context, like a stochastic automaton, (2) learns to associate different optimal actions with different contexts, much as a pattern classifier associates different class labels with different input patterns, and (3) uses a vector of parameters, $\boldsymbol{\theta}$, to compute the optimal actions and learns these optimal actions by updating the parameters using stochastic approximation procedures. The SRV algorithm is a result of these efforts. An important feature of the algorithm is that it can be implemented as a *connectionist unit* that can be incorporated into a network. Examples in which such units have been used to learn various tasks have been presented elsewhere [15, 16]. In this paper, we focus our attention mainly on the convergence properties of the SRV algorithm, although we do present supporting simulation results.

3. The SRV algorithm

Designed for associative reinforcement learning tasks defined above, the SRV unit has the structure shown in Figure 1. The interaction between the unit and the stochastic environment takes place as iterations of the following operations. Iteration n begins with the unit receiving an input $\mathbf{x}_n$ chosen randomly from a set of input vectors X. The unit uses the input $\mathbf{x}_n$ and two internal parameter vectors $\boldsymbol{\theta}_n$ and $\boldsymbol{\phi}_n$ to compute the two parameters μ_n and σ_n of the Gaussian distribution used to generate the unit's output. The mean output, μ_n, is computed as the inner product $\boldsymbol{\theta}_n^{\mathsf{T}}\mathbf{x}_n$, and the standard deviation, σ_n, is computed in two stages by first computing an expected evaluation $\hat{r}_n$ as an inner product $\boldsymbol{\phi}_n^{\mathsf{T}}\mathbf{x}_n$ and then computing σ_n as a function $s(\hat{r}_n)$ of the expected evaluation. $s(\hat{r}_n)$ is a monotonically decreasing, nonnegative function of $\hat{r}_n$. Moreover, $s(1.0) = 0.0$, so that when the maximum reinforcement is expected, the standard deviation is zero. The output of the unit, z_n, is generated as a random variable from the Gaussian distribution with parameters μ_n, and σ_n. The unit then receives an evaluative feedback $r(z_n, \mathbf{x}_n)$ from the environment, which it uses to adjust future outputs by updating the parameter vectors $\boldsymbol{\theta}_n$ and $\boldsymbol{\phi}_n$.

The evaluative signal $r(z, \mathbf{x})$ is a random variable whose distribution coincides almost everywhere with the distribution $H(r \mid z, \mathbf{x})$, which belongs to a family of distributions that depend on the parameters z and $\mathbf{x}$. Let

$$M(z, \mathbf{x}) = \int_{-\infty}^{\infty} r\,dH(r \mid z, \mathbf{x}) \tag{5}$$

be the regression function corresponding to this family of distributions (i.e., $M(z, \mathbf{x}) = E\{r \mid z, \mathbf{x}\}$). We assume that $M(.)$ is measureable and continuously differentiable almost everywhere.

The unit uses the following algorithm to update the parameter vector $\boldsymbol{\theta}$, thus generating a sequence of random vectors $\boldsymbol{\theta}_n$:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \sigma_n \left(r(z_n, \mathbf{x}_n) - \hat{r}_n\right)(z_n - \mu_n)\mathbf{x}_n, \tag{6}$$

$$\text{where } \mu_n = \boldsymbol{\theta}_n^{\mathsf{T}} \mathbf{x}_n, \tag{7}$$

$$\sigma_n = s(\hat{r}_n), \tag{8}$$

$$\hat{r}_n = \boldsymbol{\phi}_n^{\mathsf{T}} \mathbf{x}_n, \text{ and} \tag{9}$$

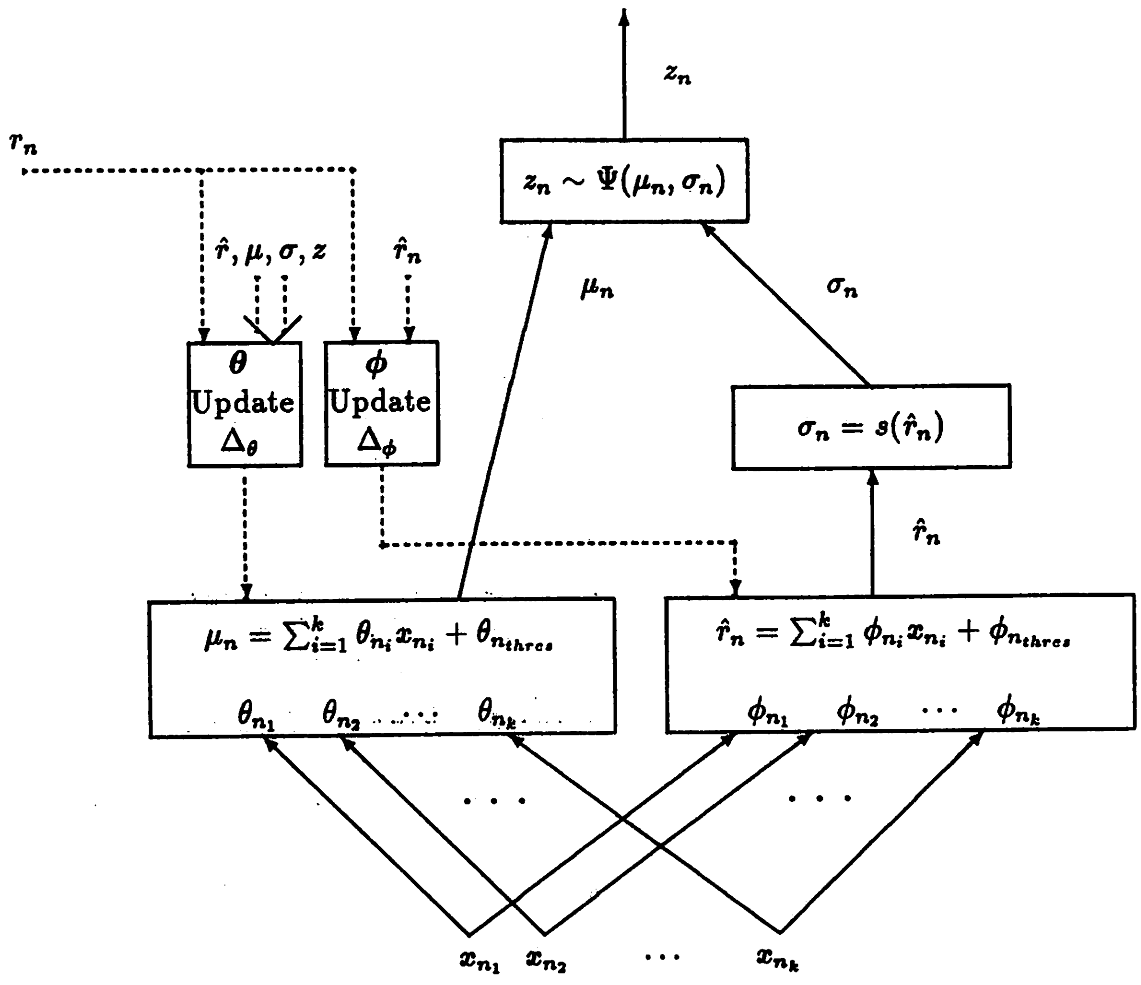

z_n
r_n
$z_n \sim \Psi(\mu_n, \sigma_n)$
$\hat{r}, \mu, \sigma, z$
$\hat{r}_n$
μ_n
σ_n
θ Update Δ_θ
ϕ Update Δ_ϕ
$\sigma_n = s(\hat{r}_n)$
$\hat{r}_n$
$\mu_n = \sum_{i=1}^{k} \theta_{n_i} x_{n_i} + \theta_{n_{thres}}$
θ_{n_1} θ_{n_2} … θ_{n_k}
$\hat{r}_n = \sum_{i=1}^{k} \phi_{n_i} x_{n_i} + \phi_{n_{thres}}$
ϕ_{n_1} ϕ_{n_2} … ϕ_{n_k}
x_{n_1} x_{n_2} … x_{n_k}

Figure 1:

$$z_n \sim N(\mu_n, \sigma_n). \tag{10}$$

(6) can be written as

$$\theta_{n+1} = \theta_n + \sigma_n^2 y_n(\theta_n, \phi_n, \mathbf{x}_n)\mathbf{x}_n \tag{11}$$

$$\text{where } y_n(\theta_n, \phi_n, \mathbf{x}_n) = (r(z_n, \mathbf{x}_n) - \hat{r}_n)\left(\frac{z_n - \mu_n}{\sigma_n}\right), \tag{12}$$

and $\mu_n, \sigma_n, \hat{r}_n,$ and z_n are as defined in (7), (8), (9), and (10) respectively.

An intuitive explanation for these update equations is the following. We can view the fraction in (12) as the *normalized noise* (or jitter) that has been added to the mean output of the unit. If this noise has caused the unit to receive an evaluation signal that is *more* than the expected evaluation, then it is desirable for the unit to have an output closer to the current output z_n. The mean output value should therefore be changed in the direction of the noise. That is, if the noise is positive, the unit should update its parameters so that the mean value increases. Conversely, if the noise is negative, the parameters should be updated so that the mean value decreases. On the other hand, if the evaluation received is *less* than the expected evaluation, then the unit should adjust its mean in the direction *opposite* to that of the noise. It can be verified that the above equations have the effect described above on the mean output.

Updating of the parameter vector ϕ used for computing the expected evaluation is relatively straightforward. We want to associate with each input vector a corresponding reinforcement value, and since both the input vector and the reinforcement value are supplied to the unit, we can use the LMS rule of Widrow & Hoff (1960) to learn this association:

$$\phi_{n+1} = \phi_n + \rho_n \left(r(z_n, \mathbf{x}_n) - \hat{r}_n \right) \mathbf{x}_n. \tag{13}$$

This completes the description of the SRV algorithm in its original form. It can be seen from the above description of the SRV algorithm that it involves a complex interaction between the two concurrent learning processes, namely, the process for learning the expected evaluation for a given input, and the process for learning the optimal output for that input. This interaction makes rigorous analysis of the learning system as a whole difficult. Since our primary interest here is in the learning of optimal outputs for given inputs, in order to make the analysis tractable, we make the following simplifications to the SRV unit and the following assumptions.

Simplifications and assumptions

S1 In the SRV algorithm, the parameter vector ϕ is used to learn an estimate of the expected evaluation $\hat{r}_n$ given the mean output μ_n for an input $\mathbf{x}_n$. As discussed above, this estimation process interacts in a complex fashion with the process of estimating the optimal output, making analysis difficult. We therefore decided to simplify the algorithm by eliminating the internal estimation of the expected evaluation. Instead, in each iteration, the unit obtains this estimate directly from the environment. It does so by emitting another output, which is the mean μ_n, and receiving an evaluation $r(\mu_n, \mathbf{x}_n) \sim H(r \mid \mu_n, \mathbf{x}_n)$ which it uses in the place of $\hat{r}_n$ in (6). The usual output z_n is also output to the environment, and the resulting evaluation $r(z_n, \mathbf{x}_n) \sim H(r \mid z_n, \mathbf{x}_n)$

is used as before in the update equation. As we no longer required it for computing $\hat{r}_n$, we discarded the parameter vector $\boldsymbol{\phi}$.

S2 Since the unit no longer computes $\hat{r}_n$ internally, it does not compute the standard deviation σ_n as a function of $\hat{r}_n$. Instead, it uses a fixed sequence of real numbers $\{\sigma_n\}$, with the properties stated below.

Given these simplifications, the equations describing the generation of the random vector sequence $\{\boldsymbol{\theta}_n\}$ are written as

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \sigma_n \left(r(z_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n)\right)(z_n - \mu_n)\mathbf{x}_n, \tag{14}$$

$$\text{where } \mu_n = \boldsymbol{\theta}_n^{\mathsf{T}} \mathbf{x}_n, \text{ and} \tag{15}$$

$$z_n \sim N(\mu_n, \sigma_n). \tag{16}$$

As before, we can define a function $y_n()$ so that (14) can be written as

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \sigma_n^2 y_n(\boldsymbol{\theta}_n, \boldsymbol{\phi}_n, \mathbf{x}_n)\, \mathbf{x}_n \tag{17}$$

$$\text{where } y_n(\boldsymbol{\theta}_n, \mathbf{x}_n) = \left(r(z_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n)\right)\left(\frac{z_n - \mu_n}{\sigma_n}\right), \tag{18}$$

and μ_n and z_n are as defined in equations 15 and 16 respectively.

The following assumptions are necessary to define the optimal output for each input in X, and to ensure that the random vector sequence $\boldsymbol{\theta}_n$ converges to the corresponding optimal parameter values:

A1 The sequence of real numbers $\{\sigma_n\}$ is such that

$$\sigma_n \geq 0, \qquad \sum_{n=1}^{\infty} \sigma_n^3 = \infty, \qquad \sum_{n=1}^{\infty} \sigma_n^4 < \infty.$$

An example of such a sequence is $\left\{1/n^{1/3}\right\}$. Note that these three conditions imply that $\sigma_n^3 \rightarrow 0$. In the limit, therefore, the standard deviation of the output of the SRV unit goes to zero (see S2 above), and the output of the unit equals the mean.

A2 1. The input vectors $\mathbf{x}_n \in X = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}\}$, a set of linearly independent vectors, and

2. $\Pr\{\mathbf{x}_n = \mathbf{x}^{(i)}\} = \xi^{(i)} > 0, \;\; 1 \leq i \leq k$.

These two conditions imply that the random input vectors are drawn from a finite set of k linearly independent vectors, with each vector in the set having a non-zero probability of being chosen at any time.

A3 For each $\mathbf{x}^{(i)}$, $1 \leq i \leq k$, there is a unique real number $\beta^{(i)} \in Z$ such that

$$M(\beta^{(i)}, \mathbf{x}^{(i)}) \;\; = \;\; \max_z M(z, \mathbf{x}^{(i)}).$$

This assumption implies that the regression function of the environmental evaluation $M(.)$ has a unique maximum for each given input $\mathbf{x}^{(i)}$. It is also assumed that $M(z, \mathbf{x})$ has bounded first and second order derivatives with respect to z. Let

$$R(z, \mathbf{x}) = \frac{\partial M(z, \mathbf{x})}{\partial z}. \tag{19}$$

A4 For each $\mathbf{x}^{(i)}$, $1 \le i \le k$,

$$\sup_{\epsilon \le |z-\beta^{(i)}| \le \frac{1}{\epsilon}} (z - \beta^{(i)})\, R(z, \mathbf{x}^{(i)}) < 0, \tag{20}$$

for all $\epsilon > 0$. This assumption says that for each i, $R(z, \mathbf{x}^{(i)})$ behaves linearly as a function of z for values of z near $\beta^{(i)}$.

A5 Finally, we assume that

$$\int_{-\infty}^{\infty} \left(r(z, \mathbf{x}^{(i)}) - M(z, \mathbf{x}^{(i)})\right)^2 dH(r \mid z, \mathbf{x}^{(i)}) \le h(1 + (z - \beta^{(i)})^2) \tag{21}$$

for some real number $h > 0$. This assumption assures that the noise in the evaluative input $r(.)$ has a bounded variance and that $\|r(z, \mathbf{x}^{(i)})\|^2$ is bounded by a quadratic function for all z and $1 \le i \le k$.

Assumptions A1, A3, A4, and A5 are fairly standard in stochastic approximation literature [22, 11]. Of these, A3 and A5 can be easily met by most evaluation functions, while assumption A4 is more restrictive, since it requires $M(.)$ to be linear in the neighborhood of the maximum for each input. It may be possible, however, to weaken this assumption as has been done in stochastic approximation literature, but only at the cost of complicating the analysis. For the sake of simplicity, we chose not to do so here.

Having delineated the above simplifications and assumptions, we are now ready to state the main result of this paper, namely, the convergence theorem for the random process defined by (14).

Theorem 1 Given assumptions A1 – A5 and a finite random initial vector $\boldsymbol{\theta}_1$, the sequence of random vectors $\{\boldsymbol{\theta}_n\}$ generated by (14) converges with probability 1 to the unique vector $\boldsymbol{\theta}^* \in \mathrm{Ra}(A) + \boldsymbol{\theta}_1$ which satisfies the equations

$$\mu_i = \boldsymbol{\theta}^{*\mathsf{T}} \mathbf{x}^{(i)} = \beta^{(i)}, \quad 1 \leq i \leq k. \tag{22}$$

This theorem says that the *mean output* μ_i of the algorithm for any given input $\mathbf{x}^{(i)}$ converges to the optimal output $\beta^{(i)}$ with probability one. But since the standard deviation used for computing the output of the algorithm tends to zero, the output converges in probability to the mean output. Hence the above theorem implies that the output of the algorithm for any given input $\mathbf{x}^{(i)}$ converges in probability to the optimal output value $\beta^{(i)}$. The proof of the theorem is given in the Appendix.

4. Discussion

A few observations of general interest can be made regarding the above strong convergence theorem and the associated assumptions and simplifications. Most of these pertain to the relationship between the original SRV algorithm and the modified version to which the convergence result applies.

1. To make rigorous analysis possible, we made simplifications S1 and S2 to the original SRV algorithm. In doing so, we eliminated the need for estimating the expected evaluations $\hat{r}$ using the parameter vector $\boldsymbol{\phi}$. In the original algorithm, however, the two

interacting processes, one for estimating the expected evaluation, and the other for estimating the optimal output, run simultaneously. Since the estimate of the expected evaluation has a critical role in the estimation of the optimal output, it is necessary to ensure the correctness of this estimate. Therefore one can expect stable convergence of the original SRV algorithm only if the output estimation process is *much slower* than the process for estimating the expected evaluation.

2. Assumption A1 places restrictions on the choice of the sequence $\{\sigma_n\}$ that are critical for the proof. Therefore this assumption cannot be discarded. But if we try to obtain an intuitive understanding of the role played by these restrictions in the convergence process, we can see that a similar effect is achieved by the method used to generate the σ_n sequence in the original SRV algorithm. The condition that $\sum_{n=1}^{\infty} \sigma_n^3 = \infty$ ensures that the sum of increments to the initial parameter vector $\boldsymbol{\theta}_1$ can be arbitrarily large, so that *any* finite $\boldsymbol{\theta}_1$ can be transformed into the optimal vector $\boldsymbol{\theta}^*$. At the same time, the condition that $\sum_{n=1}^{\infty} \sigma_n^4 < \infty$ ensures that the variance in $\boldsymbol{\theta}_n$ is finite and hence the vector cannot diverge to infinity. The σ_n sequence used in the original SRV algorithm also ensures the former consequence, since σ_n does not become zero unless the optimal outputs are being produced for all inputs. Conditions on the computation of σ_n (Equations 8 and 9) which will ensure the second property described above are currently unclear.

3. It is likely that linear independence of the input vectors, as required by A2, is not an essential condition for convergence. This is suggested by simulations of these algorithms in various tasks. We present some of these simulations in the next section. By defining

an additional evaluation criterion for the parameter vector (for example, the least-squares criterion), it may be possible to show convergence to the best approximation of optimal performance relative to this criterion.

5. Simulation results

In this section, we present simulations of the original SRV algorithm and the modified version for which the convergence result above applies in two simple associative reinforcement learning tasks. The purpose of these simulations is twofold: first, to illustrate the convergence properties of the algorithms, and second, to examine the practical importance of the simplifications and assumptions made in Section 3.. To do this, we have designed the tasks so that all the conditions of the theorem are met in the first task, while some of these are violated in the second task. Clearly, associative reinforcement learning tasks can be made very difficult to solve, for example, by increasing the number of input vectors or their dimensionality. We have deliberately chosen rather simple tasks so that the important aspects of the algorithm's performance are readily discernible.

Each task is defined by a finite set of input vectors (X) and their corresponding β values (see assumption A3). For Task 1, X has three vectors, $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(3)}$, which are linearly independent but not orthogonal. The set of input vectors defining Task 2 has the same three vectors as Task 1 plus two more. These five vectors together constitute a linearly dependent set (specifically, $\mathbf{x}^{(5)} = 0.5(\mathbf{x}^{(1)} + \mathbf{x}^{(3)})$.) The vectors in the input set for a task are all equally likely to occur at each time step. Thus $\xi^{(i)} = 1/3, \quad i = 1,2,3$ for Task 1, and $\xi^{(i)} = 0.2, \quad i = 1, \cdots, 5$ for Task 2. These conditions ensure that A2 is satisfied by the set of vectors for Task 1. We picked the initial parameter vector $\boldsymbol{\theta}_1$ (and $\boldsymbol{\phi}_1$ for the SRV algorithm) randomly for each run of the simulation.

The sequence $\{\sigma_n\}_{n=1}^{\infty}$ used by the modified SRV algorithm was defined in our simulations as

$$\sigma_n = \frac{1}{(\lfloor n/20 \rfloor)^{1/3}}, \tag{23}$$

where $\lfloor \ \rfloor$ denotes the *floor* function. This sequence satisfies the conditions of A1, as the reader can easily verify. In the case of the SRV algorithm, we first computed $\hat{r}$ as

$$\hat{r}_n = f(\boldsymbol{\phi}_n^{\mathsf{T}} \mathbf{x}_n), \tag{24}$$

where

$$f(a) = \frac{1}{1 + e^{-a}} \tag{25}$$

is the logistic function that maps the real line onto the interval $(0, 1)$. Using this, we computed σ_n for the SRV algorithm simply as

$$\sigma_n = s(\hat{r}_n) = 1 - \hat{r}_n. \tag{26}$$

In these simulations, we also set the learning rate, ρ_n, for the parameter vector $\boldsymbol{\phi}$ (see (13)) to be constant and equal to 0.5.

Two different evaluation functions, r_1 and r_2, were used for the simulations. These are defined as follows:

$$r_1(z, \mathbf{x}^{(i)}) = 1.0 - \mid f(\beta^{(i)}) - f(z) \mid, \tag{27}$$

and

$$r_2(z, \mathbf{x}^{(i)}) = N(r_1(z, \mathbf{x}^{(i)}), 0.1), \tag{28}$$

where $f(.)$ is the logistic function defined above, and $N(\mu, \sigma)$ is the Gaussian distribution function. Note that r_1 is a deterministic evaluation function, while r_2 returns a random evaluation whose mean is the evaluation returned by r_1. From these definitions, it is clear that the highest *expected* reinforcement, given an input vector, is 1.0 for either evaluation function. It is also easy to verify that both these evaluation functions satisfy assumptions A3, A4, and A5.

Performance of the two algorithms was measured by recording a smoothed reinforcement value at each time step over the course of a training run. The smoothed reinforcement at any time step was computed as the average of the reinforcement received over the last 100 consecutive time steps. This measure conveys more information about the change in performance of the algorithms *over time* than the actual reinforcement received at a time step. Perhaps a more accurate performance measure would be the expected value of the reinforcement at each time step, given the parameters and the input vector at that time step. However, because both the output and the reinforcement are continuous valued random numbers with their own distributions, obtaining a closed form expression for this expected value is not easy. The moving average of past reinforcements is a good approximation of this expected value, and one that is easily computable. A single simulation run lasted for 4500 time steps for Task 1 and 7500 time steps for Task 2. If each input vector in X were presented sequentially to the learning algorithm, these simulation durations would correspond to 1500 presentations of the entire input set for either task. For the purposes of collecting statistics, we conducted 25 runs of each algorithm in each task with each evaluation function.

A. Task 1

For Task 1, the input vectors are $\mathbf{x}^{(1)} = (1,1,1,0)^{\mathsf{T}}$, $\mathbf{x}^{(2)} = (1,1,1,1)^{\mathsf{T}}$, and $\mathbf{x}^{(3)} = (1,0,1,1)^{\mathsf{T}}$. The corresponding optimal output values are $\boldsymbol{\beta} = (-1.3862, 1.3862, 0.0)^{\mathsf{T}}$, and the learning system receives the maximum reinforcement when it responds to an input vector with the corresponding optimal output. Note that because these input vectors are linearly independent and the initial parameter vector $\boldsymbol{\theta}_1$ is finite, it follows from Lemma 1 that there exists a optimal parameter vector $\boldsymbol{\theta}^*$ that is unique for that initial parameter vector.

The results of simulating the modified SRV algorithm in Task 1 with both evaluation functions are shown in Figure 2. Graphs in the figure are plots of the smoothed reinforcement at each time step n, $1 \leq n \leq 4500$. Figure 2(a) shows results of 10 individual runs with each type of reinforcement. The average smoothed reinforcement over 25 runs for each type of reinforcement is plotted in Figure 2(b). Comparing these plots, it is clear that the algorithm performs almost as well with random reinforcement as with deterministic reinforcement. Further, the individual runs all show identical convergence behavior. Since all the assumptions A1–A5 made for the theorem are satisfied by the definition of Task 1, these plots serve to illustrate the convergence behavior assured by the theorem. For example, in one of the simulation runs with random reinforcement, the randomly chosen value for $\boldsymbol{\theta}_1$ was $(0.3163, 0.0153, -0.3275, -0.3472)^{\mathsf{T}}$. After 4500 time steps, $\boldsymbol{\theta}_{4500}$ was $(-1.0263, 1.3097, -1.6702, 2.6557)^{\mathsf{T}}$, which is close to the theoretical asymptotic value[2] $\boldsymbol{\theta}^{*} = (-1.0644, 1.3863, -1.7082, 2.7726)^{\mathsf{T}}$ *for the above value of* $\boldsymbol{\theta}_1$. The smoothed payoff at this time step was 0.9697, which is also close to the optimal value of 1.0. Obviously, more training steps would be required for closer convergence. Moreover, because σ_n tends to zero very rapidly, longer and longer training sequences become necessary for comparable reductions in the deviation from the optimal parameters.

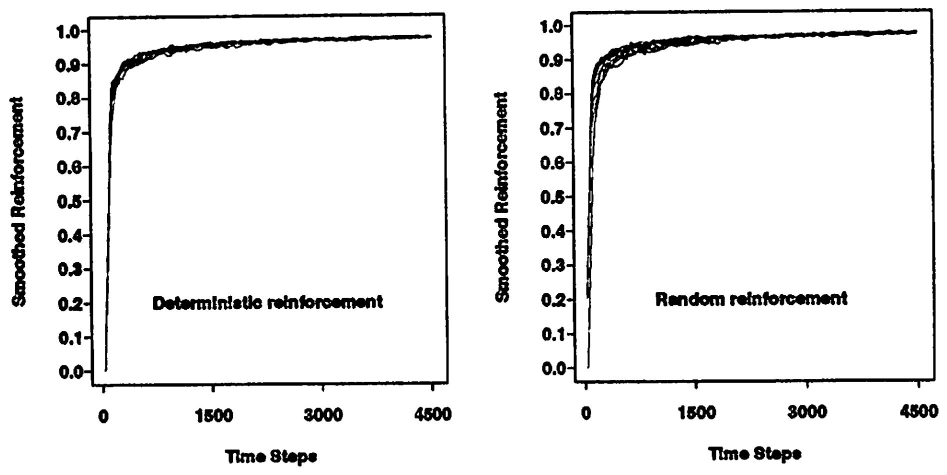
An identical set of plots of simulations of the original SRV algorithm in Task 1 are presented in Figure 3. Recall that in this version of the algorithm, an additional set of parameters is used to learn the expected reinforcement, which is then used to compute the standard deviation σ_n of the output. Because of this additional work, the SRV algorithm is slower than the modified SRV algorithm. Nevertheless, it exhibits similar convergence of the parameters to their optimal values. Further, Figure 3(b) shows that the SRV algorithm also performs equally well with both deterministic and noisy reinforcements. Thus the simplifications S1 and S2 introduced to facilitate theoretical analysis do not appear essential for convergence in practice, at least in the simple tasks presented here. Moreover, the simplifications do not seem to significantly benefit the performance in these tasks.

B. Task 2

For Task 2, the input set for Task 1 was augmented with two additional vectors. These are $\mathbf{x}^{(4)} = (1, 2, 1, 1)^{\mathsf{T}}$, and $\mathbf{x}^{(5)} = (1, 0.5, 1, 0.5)^{\mathsf{T}}$ and the corresponding optimal output values are $\beta = (2.7726, -0.6931)^{\mathsf{T}}$. As noted before, with the additional input vectors, the vectors

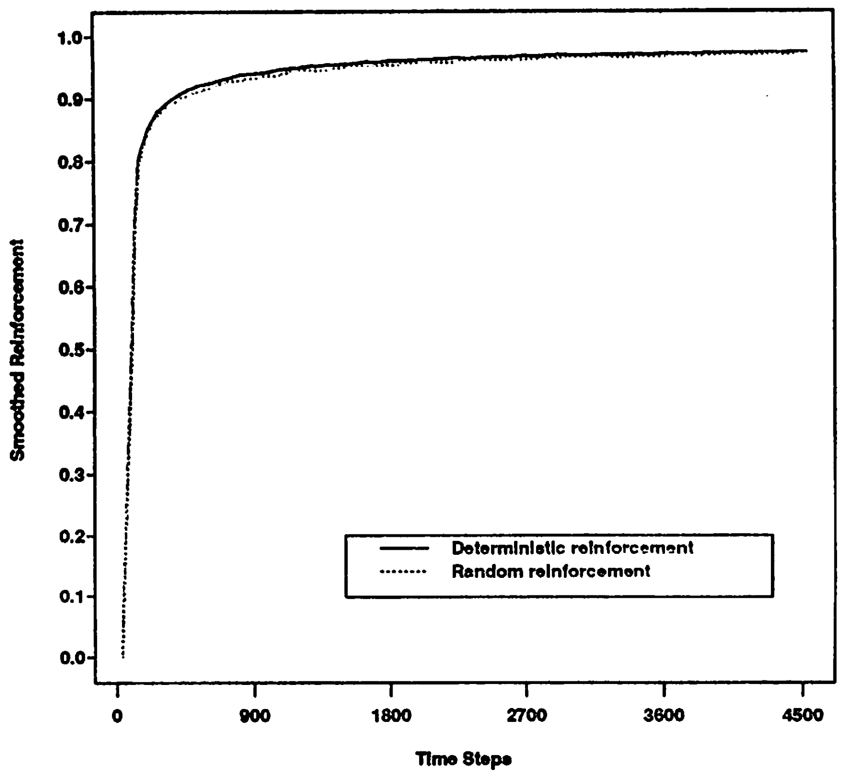
[2]The proof of Lemma 1 shows how this theoretical asymptotic value can be computed.

10 Runs of the Modified SRV Algorithm in Task 1



(a)

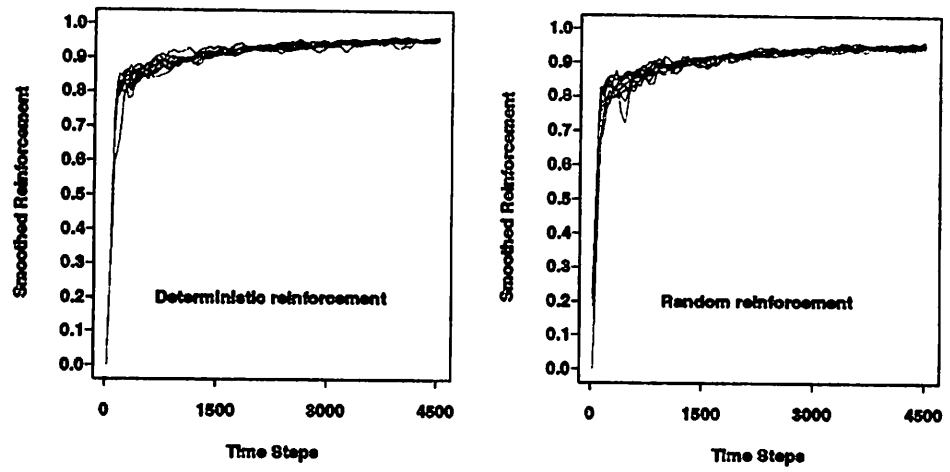
Average over 25 Runs of the Modified SRV Algorithm in Task 1



(b)

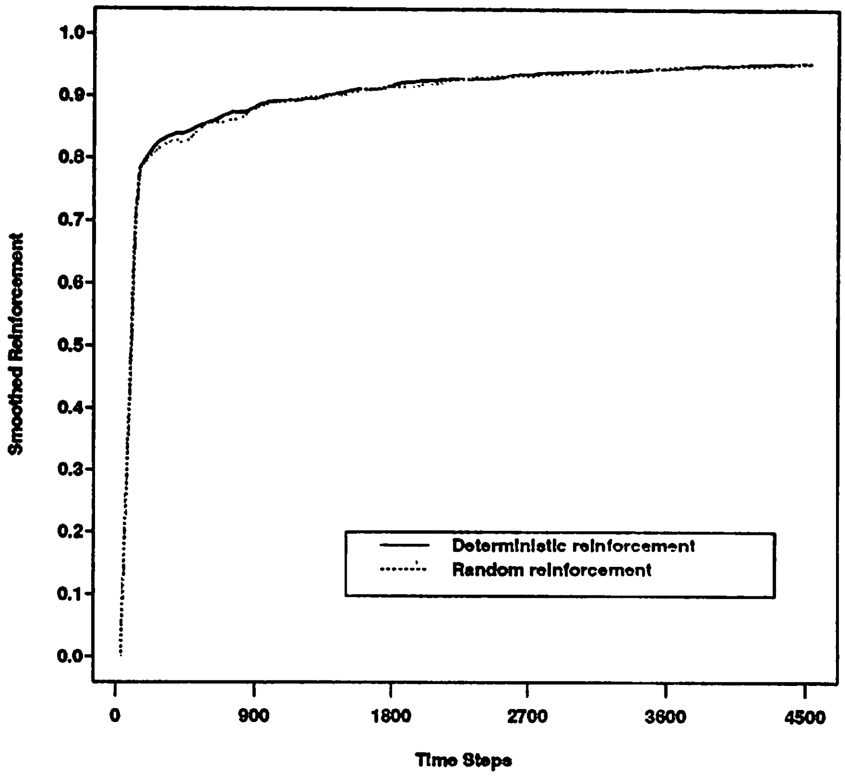
Figure 2:

10 Runs of the SRV Algorithm in Task 1



(a)

Average over 25 Runs of the SRV Algorithm in Task 1

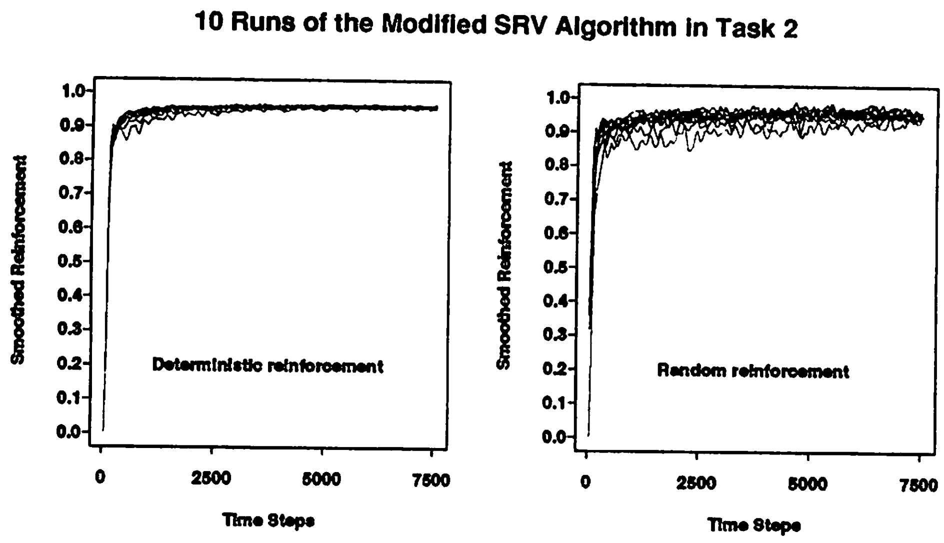


(b)

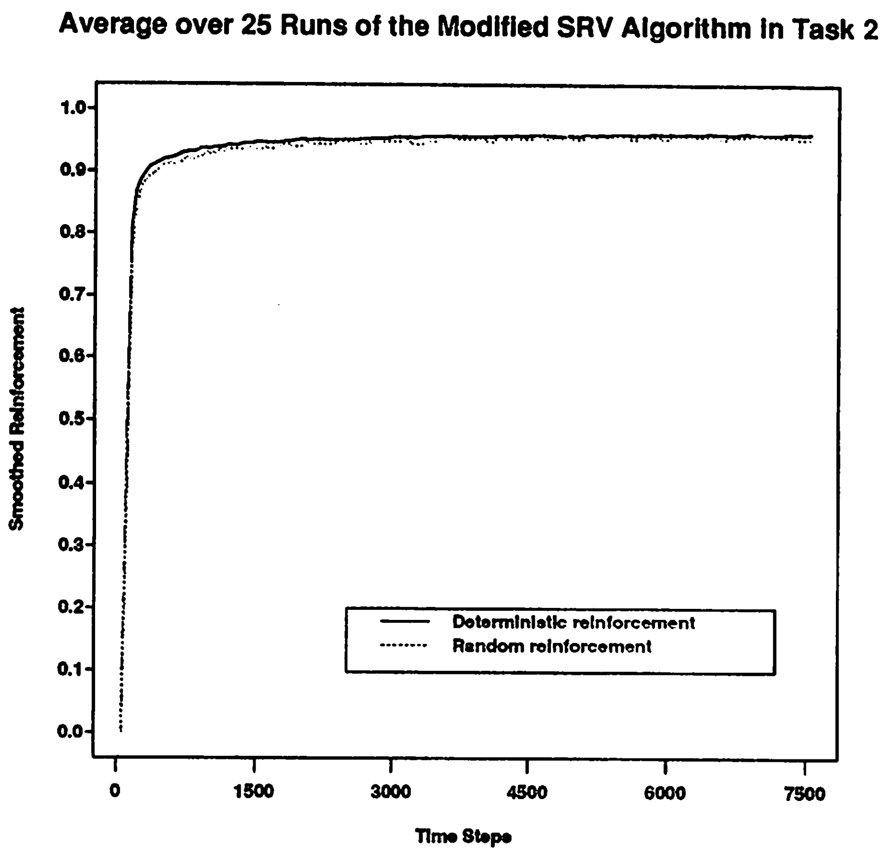
Figure 3:

in X are no longer linearly independent and hence assumption A2 is violated. Therefore it is no longer possible to use the procedure in Lemma 1 to determine the optimal parameter vector. In order to ensure that there was at least one set of parameters which would maximize the expected reinforcement, we chose the optimal output values for the new vectors in the following manner. Using the original three input vectors and their corresponding optimal output values, we computed the optimal parameter vector $\boldsymbol{\theta}^*$ assuming that $\boldsymbol{\theta}_1 = \mathbf{0}$. This yields $\boldsymbol{\theta}^* = (-1.3863, 1.3863, -1.3863, 2.7726)^{\mathsf{T}}$. We then set $\beta^{(i)} = \boldsymbol{\theta}^{*\mathsf{T}}\mathbf{x}^{(i)}$ for $i = 4, 5$. Clearly, $\boldsymbol{\theta}^*$ is the only parameter vector which can yield optimal outputs for all five inputs, and hence it is the optimal value for the parameters regardless of the starting value $\boldsymbol{\theta}_1$.

Figures 4 and 5 show the results of simulating the modified SRV algorithm and SRV algorithm respectively in Task 2. As before, part (a) of each figure depicts the performance over 10 runs with each type of reinforcement, while part (b) depicts the performance averaged over 25 runs. Comparing these figures with those for Task 1, we can see that the individual runs are more jittery, especially in the case of random reinforcement. However, all runs of the simulation converged in the case of both algorithms, as is evidenced by the plots of performance averaged over 25 runs. One run using the SRV algorithm did not start to converge until after the 2500th time step. The ostensible reason for this was an unfortunate choice of the random initial parameters for ϕ that led the algorithm to predict higher reinforcement values than were being actually obtained. Until the reinforcement predictor became more accurate, the algorithm could not learn the right outputs and converge. Observation of individual runs leads us to believe that the non-uniform convergence behavior is probably due to the random initial parameters, since, for this task, there is a single set


10 Runs of the Modified SRV Algorithm in Task 2
Smoothed Reinforcement
Deterministic reinforcement
Random reinforcement
Time Steps
0
2500
5000
7500
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

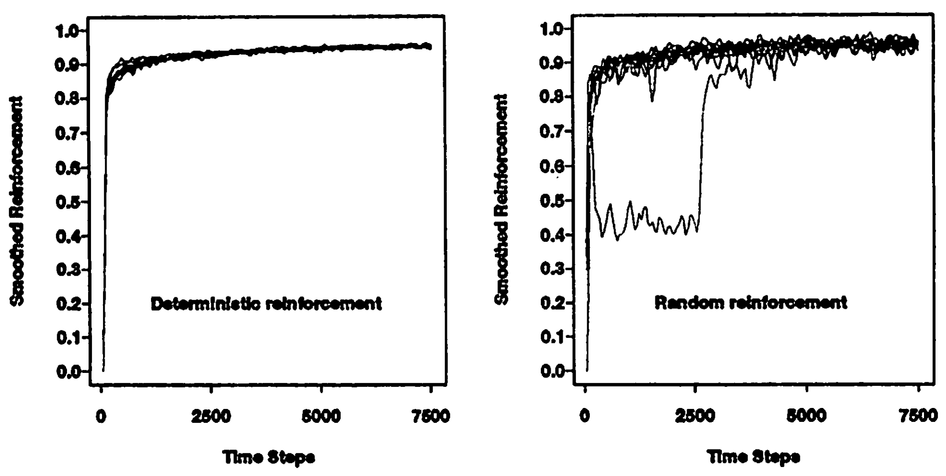
(a)


Average over 25 Runs of the Modified SRV Algorithm in Task 2
Smoothed Reinforcement
Deterministic reinforcement
Random reinforcement
Time Steps
0
1500
3000
4500
6000
7500
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

(b)

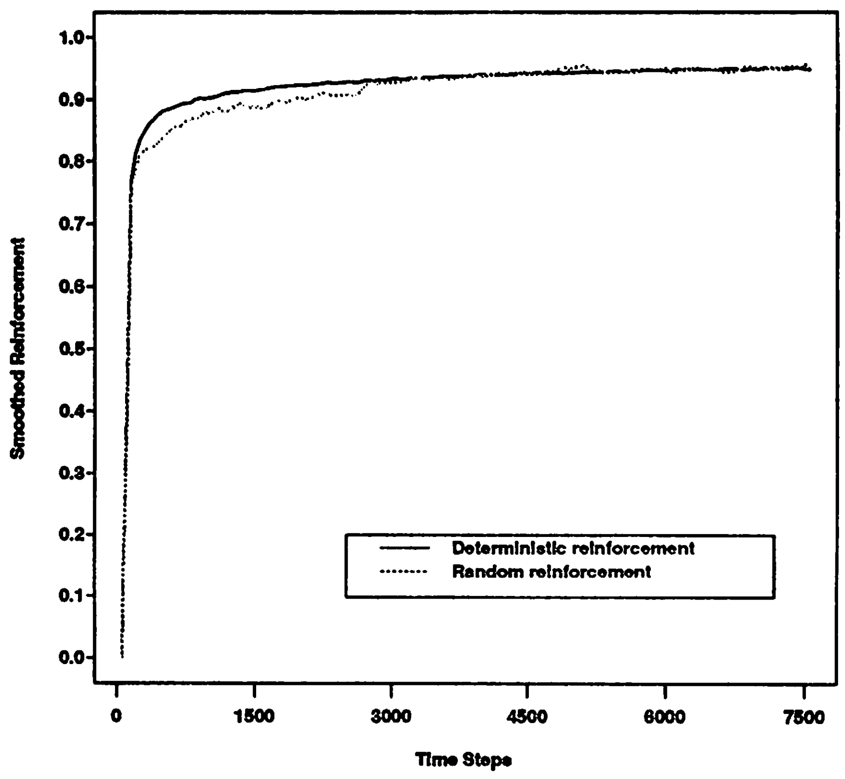
Figure 4:

10 Runs of the SRV Algorithm in Task 2



(a)

Average over 25 Runs of the SRV Algorithm in Task 2



(b)

Figure 5:

of optimal parameters independent of the initial values. This also accounts for the slower overall convergence rate for Task 2 as compared with Task 1.

Figure 4 shows that the condition that the input vectors be linearly independent is not critical for convergence of the modified SRV algorithm, as long as the task itself is linear. Using random reinforcement also appears to affect the rate of convergence only marginally. The same is true for the SRV algorithm also. Although these results only cover the case of a finite number of input vectors, elsewhere [15, 16] we have presented simulations in which the SRV algorithm performs well even when the input vectors were drawn from an infinite set. These simulations show that the SRV algorithm appears to have all the desirable convergence properties proved for the modified SRV algorithm.

6. Conclusion

In this paper, we have presented an algorithm that can learn associative maps from input vectors to actions that are real-valued without the necessity of having the desired responses available to the algorithm. This work extends the pioneering work of Barto and Anandan in synthesizing associative reinforcement learning algorithms using techniques from pattern classification and automata theory. We have also presented a strong convergence theorem for a simplified version of the algorithm operating under certain conditions. Simulation results have been used to show that neither the simplifications nor some of the conditions of the theorem are essential for good convergence in practice. We feel that the central idea of the SRV algorithm, namely, using predictions of the outcomes of actions to incrementally optimize the actions, is an important one that merits further study. Finally, the ability to learn real-valued functions using less informative training signals than conventional learning algorithms may prove to be a useful tool in several applications.

Acknowledgements

The author wishes to thank Rahul Simha for his help in the development of the results presented in this paper and A. Barto for his support and advice. The simulations reported in this paper were conducted using the X$_{\text{i}}$NG simulation package developed by J. Bachrach.

Appendix

Convergence Proof

Before proving the convergence result, we make the following observation about the algorithm and establish three supporting lemmas.[3]

Observation Let A denote the matrix whose columns are the input vectors $\mathbf{x}^{(i)}, 1 \leq i \leq k$, and let Ra($A$) denote the range of A. Then $\boldsymbol{\theta}_n \in \text{Ra}(A)+\boldsymbol{\theta}_1,\ \ n \geq 1$. This is readily apparent if we write

$$\boldsymbol{\theta}_n \quad = \quad \boldsymbol{\theta}_1 + \sum_{j=1}^{n-1} \Delta\boldsymbol{\theta}_j.$$

[3]Supporting results similar to the above observation and Lemma 1 were used by Barto and Anandan [4] in their proof of convergence of the A_{R-P} algorithm.

$$\text{where} \quad \Delta\boldsymbol{\theta}_j = \boldsymbol{\theta}_{j+1} - \boldsymbol{\theta}_j,$$

and observe from (17) that each $\Delta\boldsymbol{\theta}_j$ is a scalar times $\mathbf{x}_j \in X$.

Lemma 1 Given an arbitrary finite $\boldsymbol{\theta}_1$ and $\boldsymbol{\beta} = (\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(k)})$ as in A3, there exists a unique $\boldsymbol{\theta}^* \in \mathrm{Ra}(A) + \boldsymbol{\theta}_1$ such that

$$\mu_i = \boldsymbol{\theta}^{*\mathsf{T}} \mathbf{x}^{(i)} = \beta^{(i)}, \tag{29}$$

for i, $1 \leq i \leq k$.

Proof: Consider the equation

$$A^{\mathsf{T}} \boldsymbol{\theta} = \boldsymbol{\beta}, \tag{30}$$

where A is the matrix whose columns are the input vectors (as in the observation above). Since the rows of A are independent (from A2) there exists at least one $\boldsymbol{\theta}^*$ that satisfies (30). Also, since $\boldsymbol{\theta}^* \in \mathrm{Ra}(A) + \boldsymbol{\theta}_1$, we can rewrite (30) as $A^{\mathsf{T}}(A\boldsymbol{\omega} + \boldsymbol{\theta}_1) = \boldsymbol{\beta}$ for some $\boldsymbol{\omega} \in \Re^k$. Since the columns of A are independent, $A^{\mathsf{T}}A$ is invertible, and $\boldsymbol{\omega}$ is the unique vector

$$\boldsymbol{\omega} = (A^{\mathsf{T}}A)^{-1}(\boldsymbol{\beta} - A^{\mathsf{T}}\boldsymbol{\theta}_1). \tag{31}$$

Thus $\boldsymbol{\theta}^* = A\boldsymbol{\omega} + \boldsymbol{\theta}_1$, with $\boldsymbol{\omega}$ defined by (31), is the unique vector in $\mathrm{Ra}(A) + \boldsymbol{\theta}_1$ that satisfies (29). □

Lemma 2

$$E\{y_n(\boldsymbol{\theta}_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\} = \sigma_n R(\boldsymbol{\theta}_n^{\mathsf{T}} \mathbf{x}_n, \mathbf{x}_n). \tag{32}$$

Proof: From (18),

$$y_n(\boldsymbol{\theta}_n, \mathbf{x}_n) = (r(z_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n)) \left(\frac{z_n - \mu_n}{\sigma_n}\right),$$

where $z_n \sim N(\mu_n, \sigma_n)$ is a Gaussian random variable and $\mu_n = \boldsymbol{\theta}_n^{\mathsf{T}} \mathbf{x}_n$. Taking conditional expectations on both sides, we get

$$\begin{aligned} & E\{y_n(\boldsymbol{\theta}_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\} \\ &\quad = E\left\{r(z_n, \mathbf{x}_n)\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} - E\left\{r(\mu_n, \mathbf{x}_n)\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} \\ &\quad = E\left\{r(z_n, \mathbf{x}_n)\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} \qquad (33) \end{aligned}$$

because $r(\mu_n, \mathbf{x}_n)$ and $\left(\frac{z_n - \mu_n}{\sigma_n}\right)$ are conditionally independent, given $\boldsymbol{\theta}_n$ and $\mathbf{x}_n$, and $E\left\{\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} = 0$. But

$$\begin{aligned} & E\left\{r(z_n, \mathbf{x}_n)\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} \\ &\quad = \int_{-\infty}^{\infty} E\left\{r(z_n, \mathbf{x}_n)\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n, z_n\right\} dD_n(z_n \mid \boldsymbol{\theta}_n, \mathbf{x}_n), \end{aligned}$$

where $D_n(.)$ is the distribution function for z_n. Therefore

$$E\left\{r(z_n, \mathbf{x}_n)\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\}$$

$$= \int_{-\infty}^{\infty} M(z_n, \mathbf{x}_n) \left(\frac{z_n - \mu_n}{\sigma_n}\right) dD_n(z_n \mid \boldsymbol{\theta}_n, \mathbf{x}_n),$$

$$= E\left\{M(z_n, \mathbf{x}_n) \left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\}. \tag{34}$$

Substituting the result (34) in (33), we get

$$E\{y_n(\boldsymbol{\theta}_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\}$$

$$= E\left\{M(z_n, \mathbf{x}_n) \left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\}$$

$$= E\left\{\left[M(\mu_n, \mathbf{x}_n) + \frac{\partial M(\mu_n, \mathbf{x}_n)}{\partial z}(z_n - \mu_n) + \frac{\partial^2 M(\zeta_n, \mathbf{x}_n)}{\partial z^2}\frac{(z_n - \mu_n)^2}{2}\right]\right.$$

$$\left.\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\},$$

using a second order Taylor series expansion of $M(z_n, \mathbf{x}_n)$ about μ_n (ζ_n lies between z_n and μ_n). Therefore

$$E\{y_n(\boldsymbol{\theta}_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\}$$

$$= M(\mu_n, \mathbf{x}_n) E\left\{\left(\frac{z_n - \mu_n}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} + R(\mu_n, \mathbf{x}_n) E\left\{\left(\frac{(z_n - \mu_n)^2}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\}$$

$$+ \frac{1}{2} E\left\{\frac{\partial^2 M(\zeta_n, \mathbf{x}_n)}{\partial z^2}\left(\frac{(z_n - \mu_n)^3}{\sigma_n}\right) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\}.$$

The first term on the rhs above is zero because the odd moments of z_n, a Gaussian random variable, are zero. Since $M(.)$ has bounded second order derivatives by assumption A3, the last term on the rhs can also be seen to be zero. Hence we get

$$E\{y_n(\boldsymbol{\theta}_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n\} = \sigma_n R(\mu_n, \mathbf{x}_n)$$

$$= \sigma_n R(\boldsymbol{\theta}_n^{\mathsf{T}} \mathbf{x}_n, \mathbf{x}_n).$$

□

Lemma 3

$$E\left\{\|y_n(\boldsymbol{\theta}_n, \mathbf{x}_n)\|^2 \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} \leq K(1 + \|\boldsymbol{\theta}_n - \boldsymbol{\theta}^*\|^2). \tag{35}$$

Proof: From (18)

$$\begin{aligned}
&E\left\{\|y_n(\boldsymbol{\theta}_n, \mathbf{x}_n)\|^2 \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} \\
&\quad = E\left\{(r(z_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n))^2 \left(\frac{z_n - \mu_n}{\sigma_n}\right)^2 \mid \boldsymbol{\theta}_n, \mathbf{x}_n\right\} \\
&\quad = \int_{-\infty}^{\infty} E\left\{(r(z_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n))^2 \mid \boldsymbol{\theta}_n, \mathbf{x}_n, z_n\right\} \left(\frac{z_n - \mu_n}{\sigma_n}\right)^2 dD_n(z_n \mid \boldsymbol{\theta}_n, \mathbf{x}_n). \quad (36)
\end{aligned}$$

Now

$$\begin{aligned}
&E\left\{(r(z_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n))^2 \mid \boldsymbol{\theta}_n, \mathbf{x}_n, z_n\right\} \\
&\qquad = E\left\{r^2(z_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n, z_n\right\} + E\left\{r^2(\mu_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n, z_n\right\} \\
&\qquad\quad - 2E\left\{r(z_n, \mathbf{x}_n) r(\mu_n, \mathbf{x}_n) \mid \boldsymbol{\theta}_n, \mathbf{x}_n, z_n\right\}.
\end{aligned}$$

Defining $\beta_n = \beta^{(i)}$ if $\mathbf{x}_n = \mathbf{x}^{(i)} \in X$ (i.e., $\beta_n = \boldsymbol{\theta}^{*\mathsf{T}} \mathbf{x}_n$), we can use assumption A5 and the conditional independence a.e. of $r(z_n, \mathbf{x}_n)$ and $r(\mu_n, \mathbf{x}_n)$, given $\boldsymbol{\theta}_n$, $\mathbf{x}_n$, and z_n, in the above equation to obtain

$$\begin{aligned}
&E\left\{(r(z_n,\mathbf{x}_n)-r(\mu_n,\mathbf{x}_n))^2 \mid \theta_n,\mathbf{x}_n,z_n\right\} \\
&\quad\le [h(1+(z_n-\beta_n)^2+M^2(z_n,\mathbf{x}_n)]+[h(1+(\mu_n-\beta_n)^2+M^2(\mu_n,\mathbf{x}_n)] \qquad (37)\\
&\qquad\quad -2M(z_n,\mathbf{x}_n)M(\mu_n,\mathbf{x}_n)\\
&\quad= h(2+(z_n-\beta_n)^2+(\mu_n-\beta_n)^2)+(M(z_n,\mathbf{x}_n)-M(\mu_n,\mathbf{x}_n))^2. \qquad (38)
\end{aligned}$$

Substituting (38) in (36), we get

$$\begin{aligned}
&E\left\{\|y_n(\theta_n,\mathbf{x}_n)\|^2 \mid \theta_n,\mathbf{x}_n\right\}\\
&\quad\le 2h\int_{-\infty}^{\infty}\left(\frac{z_n-\mu_n}{\sigma_n}\right)^2 dD_n(z_n\mid\theta_n,\mathbf{x}_n)\\
&\qquad + h\int_{-\infty}^{\infty}\left((z_n-\beta_n)^2+(\mu_n-\beta_n)^2\right)\left(\frac{z_n-\mu_n}{\sigma_n}\right)^2 dD_n(z_n\mid\theta_n,\mathbf{x}_n)\\
&\qquad + \int_{-\infty}^{\infty}(M(z_n,\mathbf{x}_n)-M(\mu_n,\mathbf{x}_n))^2\left(\frac{z_n-\mu_n}{\sigma_n}\right)^2 dD_n(z_n\mid\theta_n,\mathbf{x}_n).
\end{aligned}$$

Using the fact that $z_n \sim N(\mu_n,\sigma_n)$ and expanding $M(z_n,\mathbf{x}_n)$ in a second order Taylor series about μ_n as before, we get

$$\begin{aligned}
&E\left\{\|y_n(\theta_n,\mathbf{x}_n)\|^2 \mid \theta_n,\mathbf{x}_n\right\}\\
&\quad\le 2h+h\int_{-\infty}^{\infty}[(z_n-\mu_n)^2+2(z_n-\mu_n)(\mu_n-\beta_n)+2(\mu_n-\beta_n)^2]\left(\frac{z_n-\mu_n}{\sigma_n}\right)^2 dD_n(.)\\
&\qquad+\int_{-\infty}^{\infty}\left[\frac{\partial M(\mu_n,\mathbf{x}_n)}{\partial z}(z_n-\mu_n)+\frac{\partial^2 M(\zeta_n,\mathbf{x}_n)}{\partial z^2}\frac{(z_n-\mu_n)^2}{2}\right]^2\left(\frac{z_n-\mu_n}{\sigma_n}\right)^2 dD_n(.)
\end{aligned}$$

$$= 2h + h\int_{-\infty}^{\infty}\frac{(z_n-\mu_n)^4}{\sigma_n^2}dD_n(.) + 2h(\mu_n-\beta_n)\int_{-\infty}^{\infty}\frac{(z_n-\mu_n)^3}{\sigma_n^2}dD_n(.)$$

$$+ 2h(\mu_n-\beta_n)^2\int_{-\infty}^{\infty}\frac{(z_n-\mu_n)^2}{\sigma_n^2}dD_n(.) + R^2(.)\int_{-\infty}^{\infty}\frac{(z_n-\mu_n)^4}{\sigma_n^2}dD_n(.)$$

$$+ \int_{-\infty}^{\infty}\left(\frac{\partial^2 M(\zeta_n,\mathrm{x}_n)}{\partial z^2}\right)^2\frac{(z_n-\mu_n)^6}{4\sigma_n^2}dD_n(.) + R(.)\int_{-\infty}^{\infty}\left(\frac{\partial^2 M(\zeta_n,\mathrm{x}_n)}{\partial z^2}\right)\frac{(z_n-\mu_n)^5}{\sigma_n^2}\,D_n(.)$$

Since the second order derivatives of $M(.)$ are bounded by assumption A3, these derivatives can be replaced in the last two terms of the rhs above by an upper bound S and factored out of the integrals without affecting the inequality. The integrals on the rhs are then all moments of z_n, which can be easily evaluated since z_n has a Gaussian distribution. Therefore

$$E\left\{\|y_n(\theta_n,\mathrm{x}_n)\|^2 \mid \theta_n,\mathrm{x}_n\right\}$$

$$\leq 2h + 3h\sigma_n^2 + 2h(\mu_n-\beta_n)^2 + 3R^2(.)\sigma_n^2 + \frac{15}{4}S^2\sigma_n^4$$

$$\leq H(1+(\mu_n-\beta_n)^2)$$

for some $H > 0$, by the boundedness of σ_n, $R(\mu_n,\mathrm{x}_n)$, and $\frac{\partial^2 M(.)}{\partial z^2}$. But

$$(\mu_n-\beta_n)^2 = (\theta_n^{\mathsf{T}}\mathrm{x}_n - \theta^{*\mathsf{T}}\mathrm{x}_n)^2 = ((\theta_n-\theta^*)^{\mathsf{T}}\mathrm{x}_n)^2 \leq \|\theta_n-\theta^*\|^2\,\|\mathrm{x}_n\|^2$$

by Cauchy's inequality. Substituting for $(\mu_n-\beta_n)^2$ in the above, we get (since $\mathrm{x}_n \in X$, a finite set), for some $K > 0$,

$$E\left\{\|y_n(\theta_n,\mathrm{x}_n)\|^2 \mid \theta_n,\mathrm{x}_n\right\} \leq K(1+\|\theta_n-\theta^*\|^2).$$

$\square$

We are now ready to prove Theorem 1. Our proof of this theorem is based on Gladyshev's proof of the convergence of the Robbins-Monro process [13]. Unlike the Robbins-Monro process, wherein the stochasticity is confined to the environment in which the learning system operates, (14) defines a *stochastic* system operating in a stochastic environment. Confounding between these two sources of randomness makes the analysis more complicated. Hence, although we employed the same basic proof technique based on Martingale theory as Gladyshev, his proof had to be extended in several non-trivial ways. Other alternative approaches could have been taken to arrive at the same result but we felt that the approach presented here is most easily comprehensible.

Proof of Theorem 1: Let

$$\mathbf{e}_n = (\boldsymbol{\theta}_n - \boldsymbol{\theta}^*). \tag{39}$$

Clearly, from (17),

$$\mathbf{e}_{n+1} = \mathbf{e}_n + \sigma_n^2\, y_n(\boldsymbol{\theta}_n, \mathbf{x}_n)\, \mathbf{x}_n. \tag{40}$$

Squaring and taking conditional expectations given $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$, we get

$$\begin{aligned} E\left\{\|\mathbf{e}_{n+1})\|^2 \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\} &= E\left\{\|\mathbf{e}_n)\|^2 \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\} \\ &\quad + \sigma_n^4 E\left\{\|y_n(\boldsymbol{\theta}_n, \mathbf{x}_n)\mathbf{x}_n\|^2 \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\} \\ &\quad + 2\sigma_n^2 E\left\{\mathbf{e}_n^{\mathsf{T}} y_n(\boldsymbol{\theta}_n, \mathbf{x}_n)\mathbf{x}_n \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\}. \end{aligned} \tag{41}$$

In this equation,

$$\begin{aligned}
E\left\{\|y_n(\boldsymbol{\theta}_n, \mathbf{x}_n)\mathbf{x}_n\|^2 \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\} &= \sum_{i=1}^{k} \xi_i E\left\{\|y_n(\boldsymbol{\theta}_n, \mathbf{x}^{(i)})\mathbf{x}^{(i)}\|^2 \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n, \mathbf{x}^{(i)}\right\} \\
&= \sum_{i=1}^{k} \xi_i E\left\{\|y_n(\boldsymbol{\theta}_n, \mathbf{x}^{(i)})\|^2 \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n, \mathbf{x}^{(i)}\right\} \|\mathbf{x}^{(i)}\|^2 \\
&\leq \sum_{i=1}^{k} K(1 + \|\mathbf{e}_n\|^2)\xi^{(i)}\|\mathbf{x}^{(i)}\|^2 \quad \text{(using Lemma 3)} \\
&\leq K_1(1 + \|\mathbf{e}_n\|^2), \qquad (42)
\end{aligned}$$

where $K_1 = kK \max_{1 \leq i \leq k} \xi^{(i)}\|\mathbf{x}^{(i)}\|^2$.

$$\begin{aligned}
\text{Also, } E\left\{\mathbf{e}_n^{\mathsf{T}} y_n(\boldsymbol{\theta}_n, \mathbf{x}_n)\mathbf{x}_n \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\} &= \sum_{i=1}^{k} E\left\{\mathbf{e}_n^{\mathsf{T}} y_n(\boldsymbol{\theta}_n, \mathbf{x}^{(i)})\mathbf{x}^{(i)} \mid \boldsymbol{\theta}_n, \mathbf{x}^{(i)}\right\} \xi^{(i)} \\
&= \sum_{i=1}^{k} E\left\{y_n(\boldsymbol{\theta}_n, \mathbf{x}^{(i)}) \mid \boldsymbol{\theta}_n, \mathbf{x}^{(i)}\right\} \mathbf{e}_n^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)} \qquad (43) \\
&= \sigma_n \sum_{i=1}^{k} R(\boldsymbol{\theta}_n^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})(\boldsymbol{\theta}_n - \boldsymbol{\theta}^*)^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)} \\
&= \sigma_n \sum_{i=1}^{k} R(\boldsymbol{\theta}_n^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})(\boldsymbol{\theta}_n^{\mathsf{T}}\mathbf{x}^{(i)} - \beta^{(i)})\xi^{(i)} \\
&\leq 0 \quad \text{(by assumption A4).} \qquad (44)
\end{aligned}$$

Using (42) and (44) in (41), we get

$$E\left\{\|\mathbf{e}_{n+1})\|^2 \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\} \leq \|\mathbf{e}_n\|^2 + \sigma_n^2 K_1(1 + \|\mathbf{e}_n\|^2)$$

$$= \|\mathbf{e}_n\|^2(1 + K_1\sigma_n^4) + K_1\sigma_n^4. \tag{45}$$

Let us define

$$\alpha_n = \|\mathbf{e}_n\|^2 \prod_{j=n}^{\infty}(1 + K_1\sigma_j^4) + \sum_{j=n}^{\infty} K_1\sigma_j^4 \prod_{i=j+1}^{\infty} (1 + K_1\sigma_i^4). \tag{46}$$

Then it is easy to show using (45) that

$$E\left\{\alpha_{n+1} \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\right\} \leq \alpha_n. \tag{47}$$

Taking conditional expectations for given $\alpha_1, \ldots, \alpha_n$ on both sides of the inequality (47), we find

$$E\left\{\alpha_{n+1} \mid \alpha_1, \ldots, \alpha_n\right\} \leq \alpha_n, \tag{48}$$

which shows that α_n is a non-negative supermartingale where

$$E\{\alpha_{n+1}\} \leq E\{\alpha_n\} \leq \cdots \leq E\{\alpha_1\} < \infty. \tag{49}$$

Therefore, by the martingale convergence theorem [9], α_n converges with probability 1. From this observation, the definition of α_n (Equation (46)) and assumption A1, we can conclude that $\|\mathbf{e}_n\|^2 \xrightarrow{\text{w.p.1}} \eta$, a random variable. Also, (49) and (46) together with A1 imply that

$$E\{\|\mathbf{e}_n\|^2\} < \infty. \tag{50}$$

Let us now take expectations on both sides of (41) after making substitutions using (42) and (43). We get

$$E\{\|\mathbf{e}_{n+1}\|^2\} - E\{\|\mathbf{e}_n\|^2\} \leq \sigma_n^4 K_1(1 + E\{\|\mathbf{e}_n\|^2\}) + 2\sigma_n^3 E\left\{\sum_{i=1}^{k} R(\boldsymbol{\theta}_n^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})\mathbf{e}_n^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)}\right\}. \tag{51}$$

Adding the first n of these inequalities, we get

$$E\{\|\mathbf{e}_{n+1}\|^2\} - E\{\|\mathbf{e}_1\|^2\} \le \sum_{j=1}^{n} \sigma_j^4 K_1(1+E\{\|\mathbf{e}_j\|^2\}) + 2\sum_{j=1}^{n} \sigma_j^3 E\left\{\sum_{i=1}^{k} R(\boldsymbol{\theta}_j^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})\mathbf{e}_j^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)}\right\}. \tag{52}$$

From inequality (52), using the boundedness of $E\{\|\mathbf{e}_n\|^2\}$ and assumption A1, it follows that

$$\sum_{j=1}^{\infty} \sigma_j^3 E\left\{\sum_{i=1}^{k} R(\boldsymbol{\theta}_j^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})\mathbf{e}_j^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)}\right\} > -\infty, \tag{53}$$

which implies that

$$\sum_{j=1}^{\infty} \sigma_j^3 E\left\{\sum_{i=1}^{k} -R(\boldsymbol{\theta}_j^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})\mathbf{e}_j^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)}\right\} < \infty. \tag{54}$$

Since $\sum_{j=1}^{\infty} \sigma_j^3$ diverges (A1) and, by A4 and A2, the quantity $\sum_{i=1}^{k} -R(\boldsymbol{\theta}_j^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})\mathbf{e}_j^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)}$ is non-negative, we can conclude that for some subsequence $\{n_j\}$,

$$\sum_{i=1}^{k} -R(\boldsymbol{\theta}_{n_j}^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})\mathbf{e}_{n_j}^{\mathsf{T}}\mathbf{x}^{(i)}\xi^{(i)} \to 0 \text{ w.p.1.}$$

Since each of the terms of the above sum is non-negative (A4), we have

$$R(\boldsymbol{\theta}_{n_j}^{\mathsf{T}}\mathbf{x}^{(i)}, \mathbf{x}^{(i)})\mathbf{e}_{n_j}^{\mathsf{T}}\mathbf{x}^{(i)} \to 0 \text{ w.p.1 } \forall\ 1 \le i \le k. \tag{55}$$

The fact that $\|\mathbf{e}_n\|^2 \to \eta$ w.p.1, together with assumptions A2, A3, and A4 and (55) imply that $\eta = 0$ w.p.1. Hence, $\boldsymbol{\theta}_n \to \boldsymbol{\theta}^*$ w.p.1. □

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] J. Alspector, R. B. Allen, V. Hu, and S. Satyanarayana. Stochastic learning networks and their electronic implementation. In *Proceedings of IEEE Conference on Neural Information Processing Systems – Natural and Synthetic*, Denver, CO, November 1987.

[3] R. C. Atkinson, G. H. Bower, and E. J. Crothers. *An Introduction to Mathematical Learning Theory*. Wiley, New York, 1965.

[4] A. G. Barto and P. Anandan. Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360–375, 1985.

[5] A. G. Barto and M. I. Jordan. Gradient following without back-propagation in layered networks. In *Proceedings of the IEEE First Annual Conference on Neural Networks*, pages II629–II636, San Diego, CA, 1987.

[6] A. G. Barto, R. S. Sutton, and P. S. Brouwer. Associative search network: A reinforcement learning associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, 40:201–211, 1981.

[7] R. R. Bush and W. K. Estes, editors. *Studies in Mathematical Learning Theory*. Stanford University Press, Stanford, CA, 1959.

[8] B. Chandrasekharan and D. W. C. Shen. On expediency and convergence in variable structure automata. *IEEE Transactions on Systems, Man, and Cybernetics*, SSC-4(2):52–60, 1968.

[9] J. L. Doob. *Stochastic Processes.* John Wiley and Sons, New York, 1953.

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

[11] A. Dvoretzky. On stochastic approximation. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 39–55, Berkeley and Los Angeles, 1956. University of California Press.

[12] B. G. Farley and W. A. Clark. Simulation of self-organizing systems by digital computer. *I.R.E Transactions on Information Theory*, 4:76–84, 1954.

[13] E. A. Gladyshev. On stochastic approximation. *Theory of Probability and Applications*, 10(2):275–278, 1965.

[14] V. Gullapalli. A stochastic algorithm for learning real-valued functions via reinforcement feedback. Technical Report 88-91, University of Massachusetts, Amherst, MA, 1988.

[15] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671-692, 1990.

[16] V. Gullapalli. Modeling cortical area 7a using stochastic real-valued (SRV) units. In D. S. Touretsky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, editors, *Connectionist*

Models: Proceedings of the 1990 Summer School. Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, 1991.

[17] E. Harth and E. Tzanakou. Alopex: A stochastic method for determining visual receptive fields. *Vision Research,* 14:1475–1482, 1974.

[18] R. L. Kashyap, C. C. Blaydon, and K. S. Fu. Stochastic approximation. In J. M. Mendel and K. S. Fu, editors, *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications.* Academic Press, New York, 1970.

[19] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Stat.*, 23(3), 1952.

[20] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, Englewood Cliffs, New Jersey 07632, 1989.

[21] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, 22(1):400–407, 1951.

[22] L. Schmetterer. Stochastic approximation. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability,* volume 1, pages 587–609, Berkeley and Los Angeles, 1961. University of California Press.

[23] R. S. Sutton. Ballistic bug, June 1982. Unpublished working paper.

[24] M. L. Tsetlin. *Automaton Theory and Modeling of Biological Systems.* Academic Press, New York, 1973.

[25] R. J. Williams. Reinforcement learning in connectionist networks: A mathematical analysis. Technical Report ICS 8605, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA, 1986.

[26] R. J. Williams. Reinforcement-learning connectionist systems. Technical Report NU-CCS-87-3, College of Computer Science, Northeastern University, 360 Huntington Avenue, Boston, MA, 1987.

[27] J. Wolfowitz. On the stochastic approximation method of robbins and monro. *Ann. Math. Stat.*, 25:457–461, 1952.

Manuscript received...

Vijaykumar Gullapalli

COINS Dept., LGRC

University of Massachusetts

Amherst, MA 01003

Footnotes

1. This material is based upon work supported by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526 and by the National Science Foundation under Grant ECS-8912623.

2. Informally, the activation that has the maximum expectation of reinforcement from the environment. A formal definition of the optimal action is given in Section 2.

3. The proof of Lemma 1 shows how this theoretical asymptotic value can be computed.

4. Supporting results similar to the above observation and Lemma 1 were used by Barto and Anandan [4] in their proof of convergence of the A_{R-P} algorithm.

Figure captions

1. Block diagram of a SRV unit showing the various computations being performed. The flow of signals used to compute the output of the unit is shown with solid lines, while the signals used for learning are shown with dashed lines.

2. Simulations results for the modified SRV algorithm in Task 1. (a) Curves showing the smoothed reinforcement for ten runs with each type of reinforcement. (b) Curves showing the smoothed reinforcement averaged over 25 runs with each type of reinforcement.

3. Simulations results for the SRV algorithm in Task 1. (a) Curves showing the smoothed reinforcement for ten runs with each type of reinforcement. (b) Curves showing the smoothed reinforcement averaged over 25 runs with each type of reinforcement.

4. Simulations results for the modified SRV algorithm in Task 2. (a) Curves showing the smoothed reinforcement for ten runs with each type of reinforcement. (b) Curves showing the smoothed reinforcement averaged over 25 runs with each type of reinforcement.

5. Simulations results for the SRV algorithm in Task 2. (a) Curves showing the smoothed reinforcement for ten runs with each type of reinforcement. (b) Curves showing the smoothed reinforcement averaged over 25 runs with each type of reinforcement.