Scientia Iranica, Vol. 12, No. 1, pp 14–25
© Sharif University of Technology, January 2005

SCIENTIA
IRANICA

Stochastic Optimization Using Continuous Action-Set Learning Automata

H. Beigy* and M.R. Meybodi[1]

In this paper, an adaptive random search method, based on continuous action-set learning automata, is studied for solving stochastic optimization problems in which only the noise-corrupted value of a function at any chosen point in the parameter space is available. First, a new continuous action-set learning automaton is introduced and its convergence properties are studied. Then, applications of this new continuous action-set learning automata to the minimization of a penalized Shubert function and pattern classification are presented.

INTRODUCTION

Optimization with noisy corrupted measurements is a common problem in many areas of engineering. Consider a system with measurements of $g(x,\alpha)$, where α is the parameter and x is the observation. The parameter optimization problem is defined so as to determine the optimal parameter, α^*, such that the performance function, $M(\alpha) = E[g(x,\alpha)]$, is optimized. Many efficient methods, like the steepest descent method and Newton's method, are available when gradient, ∇M, is explicitly available. Usually, due to the lack of sufficient information concerning the structure of function M or because of mathematical intractability, function M to be optimized is not explicitly known and only the noise-corrupted value of function $M(\alpha)$ at any chosen point, α, can be observed. Two important classes of algorithm are available for solving the optimization problem when only the noise-corrupted observations are available: Stochastic approximation based algorithms [1] and learning automata based algorithms [2,3].

Stochastic approximation algorithms are iterative algorithms in which the gradient of function, M, is approximated by a finite difference method and using the function evaluations obtained at points, which are chosen close to each other [1]. Learning automata are adaptive decision making devices that operate in unknown random environments and progressively improve their performance via a learning process. Learning automata are very useful for optimization of multi-modal functions when the function is unknown and only noise-corrupted evaluations are available. In these algorithms, a probability density function, which is defined over the parameter space, is used for selecting the next point. The reinforcement signal and the learning algorithm are used by learning automata for updating the probability density function at each stage. It is required that this probability density function converge to some probability density function where the optimal parameter, α^*, is chosen with probability as being as close as possible to unity. The distinguishing feature of the learning is that probability distribution of $g(x,\alpha)$ is unknown.

Methods based on stochastic approximation algorithms and learning automata represent two distinct approaches to learning problems. Though both approaches involve iterative procedures, updating at every stage is done in the parameter space in the first method, which may result in a local optimum, and in the probability space in the second method. Learning automata methods have two distinct advantages over the stochastic approximation algorithms. The first advantage is that the action space need not be a metric space because, as in stochastic approximation algorithms, the new value of the parameter is to be chosen close to the previous value. The second advantage is that the methods based on learning automata lead to global optimization, because, at every stage any element of the action-set can be chosen.

In this paper, an adaptive random search method for finding the global minimum of an unknown function is studied. In the first part of the paper, a new Continuous Action-set Learning Automaton (CALA) is introduced and its convergence behavior is stud-

*. *Corresponding Author, Department of Computer Engineering, Sharif University of Technology, Tehran, I.R. Iran.*

1. *Department of Computer Engineering, Amir Kabir University of Technology, Tehran, I.R. Iran.*

ied. A strong convergence theorem for this learning automaton is stated and proven. In the second part of the paper, algorithms are proposed, which use the proposed CALA for stochastic optimization of a penalized Shubert function and pattern classification. The proposed algorithm is a constant step size learning algorithm and needs only one function evaluation at each stage. The proposed algorithm is independent of the dimension of the parameter space of the function to be optimized.

The rest of this paper is organized as follows: In the next section a brief review of learning automata is presented. Then, a new continuous action-set learning automaton is given and its behavior is studied. After that, algorithms for stochastic optimization and the experiments are given. Finally, the paper is concluded.

LEARNING AUTOMATA

Learning Automata (LA) are adaptive decision making units that can learn to choose the optimal action from a set of actions by interaction with an unknown random environment. At each instant, n, the LA chooses an action, α_n, from its action probability distribution and applies it to the random environment. The random environment provides a stochastic response, which is called a reinforcement signal, to the LA. Then, the LA uses the reinforcement signal and learning algorithm to update the action probability distribution.

Learning automata can be classified into two main groups: Finite Action-set Learning Automata (FALA) and Continuous Action-set Learning Automata (CALA) [4]. The action-set of FALA is finite and the action probability distribution is represented by a probability vector that is updated by a learning algorithm. Several algorithms for learning optimal parameters have been developed for many discrete and continuous parameters [2,3,5-7]. When the FALA is used for solving optimization problems, one needs to discretize the parameter space, so that actions of LA can be possible values of the corresponding parameter. The accuracy of the solution is increased by choosing the finer discretization and, hence, increasing the number of actions of LA. However, increasing the number of actions leads to slow convergence of the learning algorithm. A more satisfying solution would be to employ an LA model where the action-set can be continuous, such as a subset of real line. Such a model of LA is called a continuous action-set learning automaton. Like FALA, CALA also use a probability distribution function to choose an action and the learning algorithm updates this function based on the reinforcement signal.

Continuous action-set learning automata that are presented in [8] at instant n use a normal distribution with mean, μ_n, and standard deviation, σ_n, for action probability distribution. At each instant, the CALA updates its action probability distribution by updating μ_n and σ_n. This CALA interacts with the environment through a choice of two actions at each instant. At each instant, n, the CALA chooses $\alpha_n \in \Re$ at random from its current distribution, $N(\mu_n, \sigma_n)$. Then, it gets reinforcement from the environment for the two actions: μ_n and α_n. Let these reinforcements be $\beta(\mu)$ and $\beta(\alpha)$. Then, the action probability distribution is updated as:

$$\mu_{n+1} = \mu_n + a f_1[\mu_n, \sigma_n, \alpha_n, \beta(\alpha), \beta(\mu)],$$

$$\sigma_{n+1} = \sigma_n + a f_2[\mu_n, \sigma_n, \alpha_n, \beta(\alpha), \beta(\mu)] - Ca[\sigma_n - \sigma_L], \tag{1}$$

where $f_1(.), f_2(.)$ and $\phi(.)$ are defined as below:

$$f_1(\mu, \sigma, \alpha, \beta(\alpha), \beta(\mu)) = \left[\frac{\beta(\alpha) - \beta(\mu)}{\phi(\sigma)}\right]\left[\frac{\alpha - \mu}{\phi(\sigma)}\right],$$

$$f_2(\mu, \sigma, \alpha, \beta(\alpha), \beta(\mu)) = \left[\frac{\beta(\alpha) - \beta(\mu)}{\phi(\sigma)}\right]\left[\left(\frac{\alpha - \mu}{\phi(\sigma)}\right)^2 - 1\right],$$

$$\phi(\sigma) = (\sigma - \sigma_L) I\{\sigma > \sigma_L\} + \sigma_L, \tag{2}$$

and $\sigma_L > 0, C > 0$ and $a \in (0,1)$ are parameters of the algorithm. The learning algorithm for CALA is described below. Since the updating given for σ_n does not automatically guarantee that $\sigma_{n+1} \geq \sigma_n$, a projected version of σ_n is always used, denoted by $\phi[\sigma_n]$. For this algorithm, it is shown that with an arbitrary large probability, μ_n will converge close to the optimal action and σ_n will converge close to σ_L, provided one chooses α and σ_L sufficiently small and C sufficiently large [8].

Continuous Action Reinforcement Learning Automaton (CARLA) is introduced in [9,10]. Let the action-sets of an automaton be a bounded continuous random variable defined over the interval $[\alpha_{\min}, \alpha_{\max}] \in \Re$. CARLA uses a continuous probability density function, $f(n)$, to choose its actions. It is assumed that no information about the actions is available at the start of learning and, therefore, the action probabilities have uniform distribution. CARLA updates $f(n)$ according to the following rule.

$$f(n+1) = \begin{cases} a[f(n) + (1-\beta_n)H(\alpha, \alpha_n)] & \text{if} \quad \alpha_n \in [\alpha_{\max}, \alpha_{\min}], \\ 0 & \text{otherwise}, \end{cases} \tag{3}$$

where a is a normalization factor, $\beta_n \in [0,1]$ is the reinforcement signal and $H(\alpha, r)$ is a symmetric Gaussian neighborhood function centered on $r = \alpha_n$ given by $H(\alpha, r) = \lambda e^{-\frac{1}{2}(\frac{\alpha - r}{\sigma})^2}$, where λ and σ are parameters that affect the height and the width of this neighborhood function.

Gullapalli proposed a generalized learning automaton with a continuous action-set, which uses the context input for selecting its actions and the reinforcement signal for updating its parameters [11]. Vasilakos et al. introduced a generalized learning automaton with a continuous action-set and has shown that this learning automaton finds the optimal action for each context vector [12]. In [13], a team of FALA and CALA is also used for stochastic optimization.

Learning automata have been used successfully in many applications, such as computer networks [14-16], solving NP-complete problems [17-19], capacity assignment [20,21], neural network engineering [22-25], and cellular networks [26-29] to mention a few.

A NEW CONTINUOUS ACTION-SET LEARNING AUTOMATON

In this section, a new Continuous Action-set Learning Automaton (CALA) is introduced, which will be used later in the paper for stochastic optimization. For the proposed CALA, the Gaussian distribution, $N(\mu_n, \sigma_n)$, is used for selection of actions, which is completely specified by the first and second order moments, μ_n and σ_n. The learning algorithm updates the mean and variance of the Gaussian distribution at any instant, using the reinforcement signal, β, obtained from the random environment. The reinforcement signal, $\beta \in [0,1]$, is a noise-corrupted reinforcement signal, which indicates a noise-corrupted observation of function $M(.)$ at the selected action. The reinforcement signal, β, is a random variable whose distribution function coincides almost with the distribution, $H(\beta|\alpha)$, that belongs to a family of distributions which depends on the parameter α. Let:

$$M(\alpha) = \mathrm{E}[\beta(\alpha)|\alpha] = \int_{-\infty}^{\infty} \beta(\alpha) dH(\beta|\alpha),$$

be a penalty function with bound $\mathcal{M}$ corresponding to this family of distributions. It is assumed that $M(.)$ is measurable and continuously differentiable almost everywhere. The CALA has to minimize $M(.)$ by observing $\beta(\alpha)$. Using the learning algorithm, one ideally requires that $\mu_n \rightarrow \mu^*$ and $\sigma_n \rightarrow 0$, as time tends to infinity.

The interaction between the CALA and the random environment takes place as iterations of the following operations. Iteration n begins by selection of an action, α_n, by the CALA. This action is generated as a random variable from the Gaussian distribution with parameters μ_n and σ_n. The selected action is applied to the random environment and the learning automaton receives an evaluative signal, $\beta(\alpha_n)$, which has the mean value, $M(\alpha_n)$, from the environment. Then, the learning automaton updates the parameters μ_n and σ_n. Initially, $M(.)$ is not known and it is desirable that with the interaction of the learning automaton and the random environment, μ and σ converge to their optimal values, which results in the minimum value of $M(.)$. The learning automaton uses the following rule to update parameters μ_n and σ_n, thus, generating a sequence of random variables, μ_n and σ_n:

$$\mu_{n+1} = \mu_n - a\beta(\alpha_n)\sigma_n(\alpha_n - \mu_n),$$

$$\sigma_{n+1} = f(\sigma_n), \tag{4}$$

where a is learning rate and $f(.)$ is a function that produces a sequence of $\{\sigma_n\}$ (described later). Equation 4 can be written as:

$$\mu_{n+1} = \mu_n - a\sigma_n^2 y_n(\alpha_n), \tag{5}$$

where:

$$y_n(\alpha_n) = \beta(\alpha_n)\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right). \tag{6}$$

An intuitive explanation for the above updating equations is as follows. One can view the fraction in Equation 6 as the normalized noise added to the mean. Since a, β and σ are all positive, the updating equation changes the mean value in the opposite direction of the noise. If the noise is positive, then the learning automaton should update its parameter, so that mean value increases and vice versa. Since $\mathrm{E}[\beta|\alpha]$ is close to unity when α is far from its optimal value and is close to zero when α is near to the optimal value, the learning automaton updates μ with large steps when α is far from its optimal value and with small steps when α is close to its optimal value. This causes a finer quantization of μ near its optimal value and a grain quantization for points far away from its optimal value. Thus, one can consider the learning algorithm as a random direction search algorithm with adaptive step sizes.

In what follows, the convergence of the proposed learning automaton in stationary environments is stated and proven. The convergence is proven, based on the following assumptions.

Assumption 1

The sequence of real numbers, $\{\sigma_n\}$, is such that $\sigma_n \geq 0, \sum_{n=1}^{\infty} \sigma_n^3 = \infty$ and $\sum_{n=1}^{\infty} \sigma_n^4 < \infty$.

Note that these conditions imply that $\sigma_n \rightarrow 0$ as $n \rightarrow \infty$. Therefore, in limit, σ_n of Gaussian distribution tends to zero and the action of the learning automaton becomes equal to the mean. The condition $\sum_{n=1}^{\infty} \sigma_n^3 = \infty$ ensures that the sum of increments to the initial mean, μ_0, can be arbitrarily large, so that any finite initial value of μ_0 can be transformed into the optimal value, μ^*. At the same time, the condition $\sum_{n=1}^{\infty} \sigma_n^4 < \infty$ ensures that the variance in μ_n is finite and the mean cannot diverge to infinity. Here, σ^2 is like the step size, thus, the above is the standard assumption in stochastic approximation algorithms on the step size.

Assumption 2

$M(\alpha)$ has a unique minimum at μ^*. Let $R(\alpha) = \frac{\partial M(\alpha)}{\partial \alpha}$, and $S(\alpha) = \frac{\partial M^2(\alpha)}{\partial \alpha^2}$, be the first and the second derivative of $M(\alpha)$, respectively. $M(\alpha)$ has a finite number of minima inside a compact set. $M(\alpha), R(\alpha)$ and $S(\alpha)$ have bounds $\mathcal{M}$, $\mathcal{R}$ and $\mathcal{S}$, respectively.

Note that this assumption ensures that there is an optimal action, α^*, for the learning automaton for which $\mathrm{E}[\beta(\alpha^*)]$ is minimum. Since, in limit, $\sigma_n \rightarrow 0$, then, this assumption ensures that there is an optimal mean, μ^*, for which M is minimized.

Assumption 3

$M(\alpha)$ is linear near μ^*, that is $\sup_{\varepsilon \leq |\alpha-\mu^*| \leq \frac{1}{\varepsilon}} (\alpha - \mu^*) R(\alpha) > 0$, for all $\varepsilon > 0$.

Assumptions 2 and 3 mean that the function being optimized has a unique minimum and behaves like a quadratic in the search area and near the optimum point.

Assumption 4

The noise in the reinforcement signal, $\beta(.)$, has a bounded variance, that is:

$$\mathrm{E}\left\{[\beta(\alpha) - M(\alpha)]^2\right\} \leq K_1 \left[1 + (\alpha - \mu^*)^2\right], \qquad (7)$$

for some real number, $K_1 > 0$.

Given the above assumptions, in what follows, the behavior of the proposed CALA is studied. The following theorem states the convergence of the random process defined by Equation 4. The method used to prove this theorem is similar to the methods used in stochastic approximation [1].

Theorem 1

Suppose that Assumptions 1 to 4 hold, μ_0 is finite and there is an optimal value of μ^* for μ. Then, if μ_n and σ_n are evolved according to the given learning algorithm, then, $\lim_{n\rightarrow\infty} \mu_n = \mu^*$ with probability 1.

Before the above theorem is proven, first, the following two lemmas are proven.

Lemma 1

$$\mathrm{E}\left[y_n(\alpha_n)|\mu_n\right] = \sigma_n R(\mu_n).$$

Proof

Let β_n denote $\beta(\alpha_n)$, where α_n is a random variable chosen from Gaussian distribution with mean μ_n and variance σ_n^2.

$$\mathrm{E}\left[y_n(\alpha_n)|\mu_n\right] = \mathrm{E}\left[M(\alpha_n)\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)\middle|\mu_n\right]. \qquad (8)$$

Replacing $M(\alpha_n)$ with its second order Taylor series expansion around μ_n and simplifying, one obtains:

$$\mathrm{E}\left[y_n(\alpha_n)|\mu_n\right] = M(\mu_n)\ E\left[\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)\middle|\mu_n\right]$$
$$+ R(\mu_n)\mathrm{E}\left[\left(\frac{(\alpha_n - \mu_n)^2}{\sigma_n}\right)\middle|\mu_n\right]$$
$$+ \frac{1}{2}\mathrm{E}\left[S(\xi)\left(\frac{(\alpha_n - \mu_n)^3}{\sigma_n}\right)\middle|\mu_n\right],$$

where ξ lies between α_n and μ_n. The first and last terms on the right hand side of the above equation are zero, because the odd moments of a Gaussian random variable, α_n, are zero. Since $M(.)$ has a bounded second derivative, the above equation can be written as $\mathrm{E}\left[y_n(\mu_n)|\mu_n\right] = \sigma_n R(\mu_n)$.■

Lemma 2

$$\mathrm{E}\left[y_n^2(\alpha_n)|\mu_n\right] \leq K_2\left[1 + (\mu_n - \mu^*)^2\right].$$

Proof

$$\mathrm{E}\left[y_n^2(\alpha_n)|\mu_n\right]$$
$$= \int_{-\infty}^{\infty} \mathrm{E}\left\{\left[\beta_n\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)\right]^2\middle|\mu_n\right\} dN(\alpha_n|\mu_n, \sigma_n)$$
$$= \int_{-\infty}^{\infty} \mathrm{E}\left\{\beta_n^2\middle|\mu_n\right\}\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n, \sigma_n). \qquad (9)$$

By substituting Inequality 7 in the above equation, one obtains:

$$\mathrm{E}\left[y_n^2(\alpha_n)|\mu_n\right]$$
$$\leq K_1 \int_{-\infty}^{\infty} \left[1 + (\alpha_n - \mu^*)^2 + M^2(\alpha_n)\right]$$
$$\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n, \sigma_n)$$
$$= K_1 \int_{-\infty}^{\infty} \left[1 + M^2(\alpha_n)\right]\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n, \sigma_n)$$
$$+ K_1 \int_{-\infty}^{\infty} \left[(\alpha_n - \mu_n)^2 + 2(\alpha_n - \mu_n)(\mu_n - \mu^*)\right.$$
$$\left. + (\mu_n - \mu^*)^2\right]\left(\frac{\alpha_n - \mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n, \sigma_n).$$

Since the odd moments of a Gaussian random variable are zero, the above equation can be written as:

$$\mathrm{E}\Big[y_n^2(\alpha_n)|\mu_n\Big] \leq K_1 + 3K_1\sigma_n^2 + K_1(\mu_n-\mu^*)^2 + K_1\int_{-\infty}^{\infty} M^2(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n,\sigma_n). \quad (10)$$

Replacing $M(\alpha_n)$ with its second order Taylor series expansion around μ_n and using the fact that M and its first and second derivatives are bounded, the last term of the above equation is equal to:

$$\int_{-\infty}^{\infty} M^2(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n,\sigma_n) \leq \int_{-\infty}^{\infty}\left[\mathcal{M}+\mathcal{R}(\alpha_n-\mu_n)+\mathcal{S}\frac{(\alpha_n-\mu_n)^2}{2}\right]^2 \left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n,\sigma_n), \quad (11)$$

where ξ lies between α_n and μ_n and $\mathcal{R}$ and $\mathcal{S}$ are bounds of the first and the second derivative of $M(.)$, respectively. Using the fact that the odd moments of a Gaussian random variable are zero, the above inequality can be simplified as:

$$\int_{-\infty}^{\infty} M^2(\alpha_n)\left(\frac{\alpha_n-\mu_n}{\sigma_n}\right)^2 dN(\alpha_n|\mu_n,\sigma_n) \leq \mathcal{M}^2 + 3\mathcal{R}^2\sigma_n^2 + 3\mathcal{M}\mathcal{S}\sigma_n^2 + \frac{15}{4}\mathcal{S}^2\sigma_n^4.$$

Substituting the above equation in Equation 10, one obtains:

$$\mathrm{E}\left[y_n^2(\alpha_n)|\mu_n\right] \leq K_1 + 3K_1\sigma_n^2 + K_1(\mu_n-\mu^*)^2 + K_1\mathcal{M}^2 + 3K_1\mathcal{R}^2\sigma_n^2 + 3K_1\mathcal{M}\mathcal{S}\sigma_n^2 + \frac{15}{4}K_1\mathcal{S}^2\sigma_n^4.$$

Since $\sigma_n, \mathcal{M}, \mathcal{R}$ and $\mathcal{S}$ are bounded random variables, there is a constant $K_2 > 0$, such that the above inequality can be written as:

$$\mathrm{E}\left[y_n^2(\alpha_n)|\mu_n\right] \leq K_2\left(1+(\mu_n-\mu^*)^2\right). \quad (12)$$

■

Proof of Theorem 1

Let $e_n = \mu_n - \mu^*$. Then using Equation 5, e_n can be defined recursively as:

$$e_{n+1} = e_n - a\sigma_n^2 y_n(\alpha_n). \quad (13)$$

Squaring, taking the conditional expectation given in $\mu_0, \cdots, \mu_n$ of both sides of the recursive formula for e_{n+1} and, then, using Lemmas 1 and 2, one obtains:

$$\begin{aligned}\mathrm{E}\Big[e_{n+1}^2|\mu_0,\cdots,\mu_n\Big] &= \mathrm{E}\left[\left(e_n - a\sigma_n^2 y_n(\alpha_n)\right)^2\Big|\mu_0,\cdots,\mu_n\right] \\ &\leq e_n^2 + a^2K_2\sigma_n^4(1+e_n^2) - 2a^2K_2\sigma_n^3 e_n R(\mu_n) \\ &\leq e_n^2 + a^2K_2\sigma_n^4(1+e_n^2) \\ &= e_n^2\left(1+a^2K_2\sigma_n^4\right) + a^2K_2\sigma_n^4. \end{aligned} \quad (14)$$

Let:

$$Z_n = e_n^2\prod_{j=n}^{\infty}\left(1+a^2K_2\sigma_j^4\right) + a^2K_2\sum_{j=n}^{\infty}\sigma_j^4\prod_{i=j+1}^{\infty}\left(1+a^2K_2\sigma_i^4\right). \quad (15)$$

Then, using Equation 15, it is easy to show that $e_n^2 \leq Z_n$ and $\mathrm{E}\{Z_{n+1}|\mu_0,\cdots,\mu_n\} \leq Z_n$. Taking the conditional expectations given, $Z_1,\cdots,Z_n$, on both sides of the above inequality, one obtains $\mathrm{E}\{Z_{n+1}|Z_0,\cdots,Z_n\} \leq Z_n$, which shows that Z_n is a non-negative super-martingale. Thus, one has $\mathrm{E}\{Z_{n+1}\} \leq \mathrm{E}\{Z_n\} \leq \cdots \leq \mathrm{E}\{Z_1\} \leq \infty$. Therefore, using the martingale convergence theorems [30], Z_n converges with probability 1. Since $e_n^2 \leq Z_n$, hence one concludes that e_n^2 converges to η with probability 1, where $\eta < \infty$ is a random variable. Taking expectation on both sides of Equation 14, one obtains:

$$\mathrm{E}\left[e_{n+1}^2\right] - \mathrm{E}\left[e_n^2\right] \leq a^2K_2\sigma_n^4\left(1+\mathrm{E}\left[e_n^2\right]\right) - 2a^2K_2\sigma_n^3 e_n R(\mu_n).$$

Adding the first n of these inequalities, one gets:

$$\mathrm{E}\left[e_{n+1}^2\right] - \mathrm{E}\left[e_1^2\right] \leq a^2K_2\sum_{j=1}^{n}\sigma_n^4\left(1+\mathrm{E}\left[e_j^2\right]\right) - 2a^2K_2\sum_{j=1}^{n}\sigma_j^3 e_j R(\mu_j).$$

Adding $\mathrm{E}\left[e_1^2\right]$ to both sides of the above inequality, one obtains:

$$\mathrm{E}\left[e_{n+1}^2\right] \leq \mathrm{E}\left[e_1^2\right] + a^2K_2\sum_{j=1}^{n}\sigma_n^4\left(1+\mathrm{E}\left[e_j^2\right]\right) - 2a^2K_2\sum_{j=1}^{n}\sigma_j^3 e_j R(\mu_j).$$

Since $\mathrm{E}\left[e_{n+1}^2\right]$ is positive, the above inequality becomes:

$$\mathrm{E}\left[e_1^2\right]+a^2K_2\sum_{j=1}^{n}\sigma_n^4\left(1+\mathrm{E}\left[e_j^2\right]\right)-2a^2K_2\sum_{j=1}^{n}\sigma_j^3 e_j R(\mu_j)\geq 0.$$

From the above inequality, using the boundness of $\mathrm{E}\left[e_j^2\right]$ (for $j>0$) and Assumption 1, it follows that:

$$2a^2K_2\sum_{j=1}^{n}\sigma_j^3 e_j R(\mu_j)\leq \mathrm{E}\left[e_1^2\right]+\sum_{j=1}^{n}a^2K_2\sigma_n^4\left(1+E\left[e_j^2\right]\right)$$

$$<\infty.$$

Since $\sum_{j=1}^{\infty}\sigma_j^3$ diverges and, by Assumption 3, the quantity of $e_jR(\mu_j)$ is positive, one can conclude that for some sequence, $\{n_j\}$, one has $e_{n_j}R(\mu_{n_j})\to 0$, with probability 1. The fact that e_n^2 converges with probability 1 to some random variable, η, together with Assumptions 1 through 4 and the above equation, it implies that $\eta=0$ with probability 1. Hence, μ_n converges to μ^* with probability 1.

NUMERICAL EXAMPLES

In this section, the application of the proposed continuous action-set learning automaton is studied. Two problems are studied: The first problem is to find a minimum of the penalized Shubert function, when the function evaluations are corrupted by noise, and the second problem is to find an optimal discriminant function for pattern classifications. In both problems, only the function evaluations are assumed to be available.

Optimization of a Function

In this section, an algorithm is given, based on the proposed continuous action-set learning automaton, for optimization of an unknown function. The proposed algorithm can be easily extended to multivariate functions by using a cooperative game of CALA with identical payoff. Consider a function $F:\Re\to\Re$ to be minimized. At any instant, n, the automaton chooses an action, α_n, using the Gaussian distribution, $N(\mu_n,\sigma_n)$, where μ_n and σ_n are the mean and the standard deviation, respectively. As before, let $\beta(\alpha)$ denote the noisy function evaluation at point $\alpha\in\Re$, such that $\beta(\alpha)=\frac{f(\alpha)-\lambda_2}{\lambda_1}$, where λ_1 and λ_2 are two appropriate scaling constants, such that $\beta(a)\in[0,1]$ and $f(.)$ is a noisy evaluation of $F(a)$. This is because of the assumption that β is a bounded and nonnegative random variable. The signal $\beta(\alpha)$ is used to update μ of the Gaussian distribution (Equation 4).

In order to study the behavior of the proposed algorithm, one must consider the optimization of the penalized Shubert function, which is borrowed from [8], when the function evaluations are noisy. The penalized Shubert function (Figure 1) is frequently used as one of the benchmark problems for global optimization problems. The penalized Shubert function is given below:

$$F(x)=\sum_{i=1}^{5} i\cos((i+1)x+1)+u(x,10,100,2), \tag{16}$$

where u is the penalizing function, given by:

$$u(x,b,k,m)=\begin{cases}k(x-b)^m & x>b\\ 0 & |x|\leq b\\ k(-x-b)^m & x<-b\end{cases}. \tag{17}$$

This function has 19 minima within interval $[-10,10]$, where three of them are global minima. The global minimum value of the penalized Shubert function is approximately equal to -12.87 and is attained at points close to -5.9, 0.4 and 6.8.

The proposed continuous action-set learning automaton and the automaton proposed in [8], with different initial values of the mean and variance parameters, are used for finding the minimum of the penalized Shubert function. The algorithm given in [8] is used with the following values of parameters: The penalizing constant, C, is equal to 5, the lower bound for the variance, $\sigma_L=0.01$ and the step size $a=2\times10^{-4}$. The proposed algorithm is used with $a=0.01$ and the variance is updated according to the following equation:

$$\sigma_n=\frac{1}{\lfloor\frac{n}{10}\rfloor^{1/3}},$$

where $\lfloor.\rfloor$ denotes the floor function. It can be easily verified that sequence $\{\sigma_n\}$ satisfies the conditions of Assumption 1. From the definition of the reinforcement

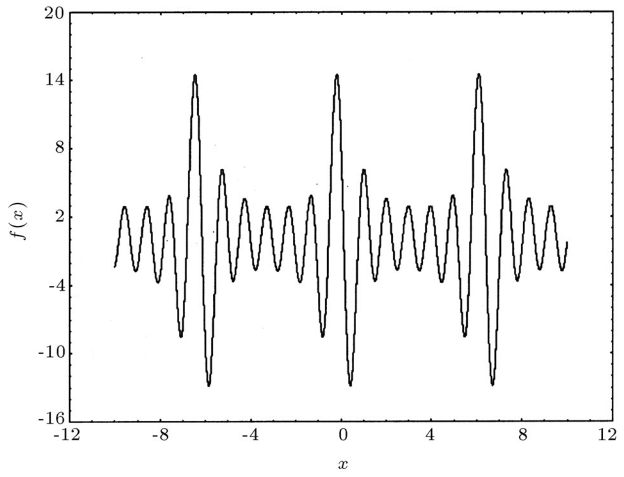

Figure 1. The penalized Shubert function.

Table 1. The simulation results with noise $U[-0.5, 0.5]$ added to the penalized Shubert function evaluations.

Case	Initial Values		Proposed Algorithm			Old Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.716046	0.1	-12.861304	2.534	0.01	-3.578
2	4	10	0.436827	0.1	-12.848987	0.4308	0.01	-12.87
3	8	5	7.805426	0.1	-3.5772	5.364	0.01	-8.5
4	8	3	6.704283	0.1	-12.868272	6.72	0.01	-12.87
5	12	6	5.460669	0.1	-8.476369	1.454	0.01	-3.58
6	-10	5	-8.107043	0.1	-3.749682	-7.1	0.01	-8.5
7	-10	6	-7.09686	0.1	-8.507154	-5.8	0.01	-12.87

signal β, it is clear that the highest expected reinforcement is 1.0 and the lowest expected reinforcement is 0.0 for either evaluation. It is easy to see that $M(\alpha) = \text{E}[\beta|\alpha]$ and β satisfy Assumptions 2 through 4. The function evaluations are corrupted by a zero mean noise randomly generated from uniform and normal distributions. The results of simulation for two algorithms, when noise is in the range [-0.5,0.5] with uniform distribution, are shown in Table 1, i.e. $f(x) = F(x) + U[-0.5, 0.5]$, where $U[a, b]$ is a random number in interval $[a, b]$ with uniform distribution. The results of simulation for two algorithms when noise has normal distribution are shown in Tables 2 through 11, i.e. $f(x) = F(x) + N(0, \sigma)$, where $N(0, \sigma)$ is a random number with uniform distribution with zero mean and the variance of σ. The simulation results show that the mean values of Gaussian distribution used in CALA always converge to a minimum of the penalized Shubert function within the interval [-10,10], which has a higher rate of convergence than the algorithm given in [8].

By comparing the results of simulations for both algorithms, one concludes that two algorithms have approximately the same accuracy, but the proposed algorithm has a higher speed of convergence. Figures 2 and 3 show the changes in μ versus n for typical runs for the proposed algorithm and the algorithm given in [8], when noise with uniform or normal distribution is added to the function evaluations. These figures show that the proposed algorithm converges to a minimum quickly and has oscillations around the minimum point. However, such oscillations could be decreased by using a small step size, which results in decreasing the rate of convergence.

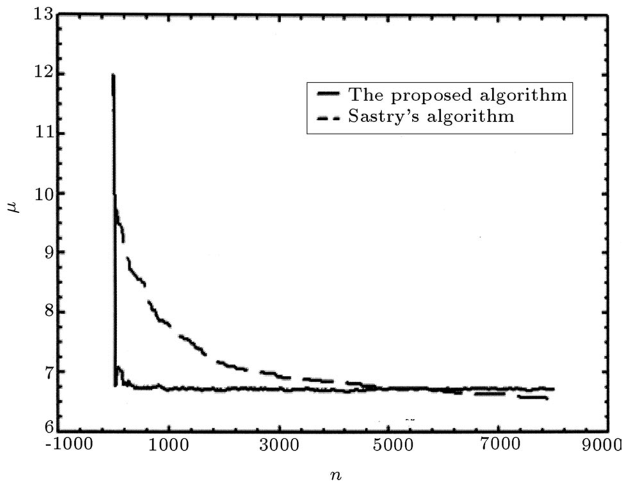

Figure 2. The convergence of the mean value of the Gaussian distribution for uniform noise.

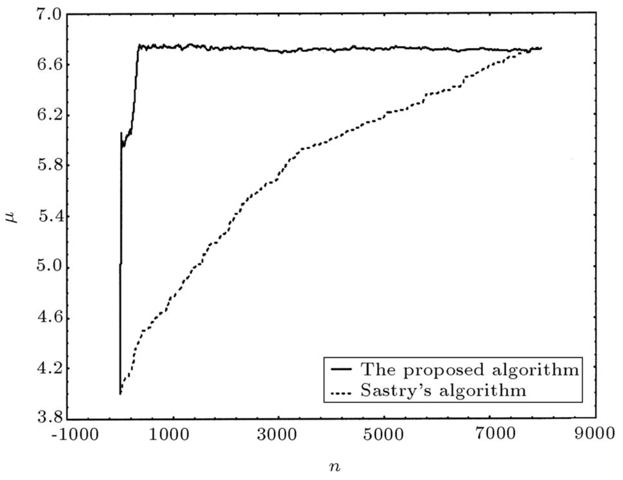

Figure 3. The convergence of the mean value of the Gaussian distribution for noise with normal distribution.

Pattern Classification

In this section, an algorithm, based on the proposed continuous action-set learning automaton for solving pattern classification problems is given. A pattern classification problem can be formulated as follows: There is a set of patterns that must be classified into a finite number of classes. The information about a pattern is summarized by a d-dimensional vector, $X = [x_1, x_2, \cdots, x_d]^T$, called a feature vector. Let m possible pattern classes be $\omega_1, \omega_2, \cdots, \omega_m$. Let $P(\omega_i)$ denote a priori probability and $P(X|\omega_i)$ (for

Table 2. The simulation results with white noise $N(0, 0.1)$ added to the penalized Shubert function evaluations.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.725683	0.107958	-12.82271	6.735791	2.204917	-12.75081
2	4	10	6.727295	0.107958	-12.813384	9.213571	0.624627	2.763267
3	8	5	-5.84428	0.107958	-12.840563	10	0.038636	-0.002154
4	8	3	0.45588	0.107958	-12.720715	9.561054	0.188162	-0.731982
5	12	6	-5.86847	0.107958	-12.853415	10	0.014074	-0.002154
6	-10	5	6.714694	0.107958	-12.864359	-9.610859	0.251364	2.879572
7	-10	6	-5.86368	0.107958	-12.865793	-9.612072	0.085958	2.875568

Table 3. The simulation results with white noise $N(0, 0.2)$ added to the penalized Shubert function evaluations.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	4.449063	0.107958	-3.739168	6.774669	2.23965	-12.185001
2	4	10	6.729869	0.107958	-12.79681	10	0.059367	-0.002154
3	8	5	0.427261	0.107958	-12.87015	10	0.023405	-0.002154
4	8	3	6.696855	0.107958	-12.84972	10	0.023574	-0.002154
5	12	6	-8.10426	0.107958	-3.749953	9.999982	0.021554	-0.002484
6	-10	5	6.704349	0.107958	-12.86835	-9.630264	0.261474	2.795851
7	-10	6	-5.84628	0.107958	-12.84872	-9.624987	0.234517	2.822746

Table 4. The simulation results with white noise $N(0, 0.3)$ added to the penalized Shubert function evaluations.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	5.46562	0.1079	-8.4960	6.6935	2.103081	-12.8358
2	4	10	6.70878	0.1079	-12.8701	10	0.037497	-0.00215
3	8	5	-8.1126	0.1079	-3.7457	10	0.04331	-0.00215
4	8	3	-5.8514	0.1079	-12.8638	10	0.017517	-0.00215
5	12	6	-7.1112	0.1079	-8.4467	10	0.021286	-0.00215
6	-10	5	6.72450	0.1079	-12.829	-9.6049	0.200885	2.898331
7	-10	6	-5.8533	0.1079	-12.8673	-9.6325	0.2594	2.783257

Table 5. The simulation results with white noise $N(0, 0.4)$ added to the penalized Shubert function evaluations

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	2.50023	0.107958	-2.639112	6.728462	2.145234	-12.806131
2	4	10	4.461055	0.107958	-3.74998	10	0.04247	-0.002154
3	8	5	6.699889	0.107958	-12.859456	9.999959	0.02963	-0.002901
4	8	3	6.726413	0.107958	-12.818587	10	0.01885	-0.002154
5	12	6	-5.842592	0.107958	-12.832664	10	0.02052	-0.002154
6	-10	5	-8.114598	0.107958	-3.743393	-9.615798	0.10445	2.862235
7	-10	6	-9.088435	0.107958	-2.728742	-9.622422	0.15321	2.834708

Table 6. The simulation results with white noise $N(0, 0.5)$ added to the penalized Shubert function evaluations.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	4.458778	0.107958	-3.749498	6.697289	2.232019	-12.851302
2	4	10	0.426808	0.107958	-12.870432	9.999987	0.02776	-0.002397
3	8	5	6.716193	0.107958	-12.860938	10	0.027162	-0.002154
4	8	3	6.693639	0.107958	-12.836154	9.692181	0.064142	-2.462594
5	12	6	0.42047	0.107958	-12.867394	10	0.015932	-0.002154
6	-10	5	-5.85382	0.107958	-12.868017	-9.604689	0.116198	2.897376
7	-10	6	6.710624	0.107958	-12.870028	-9.634266	0.259708	2.773433

Table 7. The simulation results with white noise $N(0, 0.6)$.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	5.480349	0.089604	-8.517418	4.055911	20.009987	2.400759
2	4	10	0.435484	0.089604	-12.85371	4.054561	20.00997	2.415684
3	8	5	6.726655	0.089604	-12.81718	8.006934	20.009995	-0.62230
4	8	3	6.709202	0.089604	-12.87075	8.003345	20.009943	-0.70354
5	12	6	9.756602	0.089604	-2.727776	9.999975	20.009989	-0.00260
6	-10	5	-9.98241	0.089604	-2.205438	-9.93990	20.009995	-1.70987
7	-10	6	-7.09524	0.089604	-8.510625	-9.94036	20.009991	-1.71605

Table 8. The simulation results with white noise $N(0, 1.3)$.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.70339	0.089604	-12.8669	4.05304	20.009989	2.432262
2	4	10	0.43157	0.089604	-12.8642	4.05334	20.009985	2.428998
3	8	5	6.72627	0.089604	-12.8193	8.01151	20.009995	-0.51832
4	8	3	6.72909	0.089604	-12.8020	7.99121	20.009989	-0.97549
5	12	6	9.76507	0.089604	-2.72765	9.99998	20.009995	-0.00251
6	-10	5	-9.9842	0.089604	-2.22347	-9.9406	20.009995	-1.72048
7	-10	6	-9.9838	0.089604	-2.21967	-9.9455	20.009995	-1.78518

Table 9. The simulation results with white noise $N(0, 1.9)$.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.731426	0.089604	-12.78577	4.05399	20.009993	2.421903
2	4	10	6.685631	0.089604	-12.78776	4.05446	20.009977	2.416788
3	8	5	6.723327	0.089604	-12.83486	8.00724	20.009989	-0.61527
4	8	3	6.715631	0.089604	-12.86230	7.99826	20.009981	-0.81798
5	12	6	9.760613	0.089604	-2.728751	9.99999	20.009993	-0.00232
6	-10	5	-9.98213	0.089604	-2.202685	-9.9434	20.009998	-1.75658
7	-10	6	-9.98342	0.089604	-2.215217	-9.9444	20.009991	-1.77056

Table 10. The simulation results with white noise $N(0, 2.6)$.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	5.47063	0.089604	-8.50962	4.054261	20.00998	2.418973
2	4	10	5.45497	0.089604	-8.44608	4.053005	20.00997	2.432638
3	8	5	6.70863	0.089604	-12.8708	8.005559	20.01000	-0.65347
4	8	3	6.72332	0.089604	-12.8348	8.004089	20.00998	-0.68673
5	12	6	9.77867	0.089604	-2.71016	9.999991	20.00999	-0.00231
6	-10	5	-9.9827	0.089604	-2.20913	-9.94208	20.01001	-1.73909
7	-10	6	-5.8538	0.089604	-12.8679	-9.94117	20.00996	-1.72688

Table 11. The simulation results with white noise $N(0, 3.2)$.

Case	Initial Values		Proposed Algorithm			Sastry's Algorithm		
	μ_0	σ_0	μ_{8000}	σ_{8000}	$F(\mu_{8000})$	μ_{8000}	σ_{8000}	$F(\mu_{8000})$
1	4	6	6.703756	0.089604	-12.8675	4.056128	20.010014	2.398347
2	4	10	5.465978	0.089604	-8.49724	4.054752	20.01	2.413582
3	8	5	6.711122	0.089604	-12.8696	8.011147	20.010033	-0.52657
4	8	3	6.725053	0.089604	-12.8261	7.995277	20.009974	-0.88493
5	12	6	5.471429	0.089604	-8.51119	9.999896	20.010052	-0.00404
6	-10	5	-8.10843	0.089604	-3.74913	-9.93915	20.010021	-1.69975
7	-10	6	-9.98857	0.089604	-2.26330	-9.94576	20.009981	-1.78738

$i = 1, \cdots, m$) be the class conditional densities. The function of a pattern classifier is to assign the correct class membership to each given feature vector, X. Such an operation can be interpreted as a partition of the d-dimensional space into m mutually exclusive regions. The partition boundary can be expressed in terms of discriminant functions. The forms of discriminant functions are assumed to be known, except for some parameter vectors W^i (for $i = 1, 2, \cdots, m$), which are to be identified. Associated with each class, ω_i, a discriminant function, $d_i(X, W^i)$ (for $i = 1, 2, \cdots, m$), is selected, such that if pattern X were from class ω_i, then, one would have:

$$d_i(X, W^i) > d_j(X, W^j), \qquad \forall j \neq i, \tag{18}$$

where W^i is the parameter vector for discriminant function d_i.

In this example, the learning pattern classification for a 2-class 2-dimension pattern classification problem is considered. For this problem, one discriminant function, denoted by $d(X, W)$ is sufficient. In order to find parameters of the discriminant function, W, a cooperative game of continuous action-set learning automata with identical payoff is used. In the proposed method, the teacher (environment) chooses a pattern, X, randomly based on a priori probabilities. The statistical properties of the pattern classes are assumed to be unknown to the algorithm. The team of learning automata chooses their actions, which result in the determination of parameter vector W. The parameter vector W, together with feature vector X, determine the classification. The decision rule for pattern classification is formulated as:

$$\omega = \begin{cases} \omega_1 & \text{if } d(X, W) > 0 \\ \omega_2 & \text{otherwise} \end{cases}, \tag{19}$$

where $X = [x_1, x_2]^T$ is the feature vector. The only feedback to the team of continuous action-set learning automata is the reinforcement signal, which is computed according to the following rule:

$$\beta = \begin{cases} 1 & \text{if } \omega \neq \hat{w} \\ 0 & \text{otherwise} \end{cases}, \tag{20}$$

where ω is the classification of feature vector X. The objective is to minimize the probability of misclassification of pattern X (E$[\beta|X]$), where the expectation is with respect to statistical distribution on the pattern classes. Note that this criterion is a function of the parameter vector, W, which is to be identified.

Table 12. The simulation results for pattern classification problem.

Case	Initial Values							#Iterations	Sastry's Algorithm		Proposed Algorithm	
	μ_0^m	σ_0^m	μ_0^x	σ_0^x	μ_0^b	σ_0^b	Error		Training Error	Test Error	Training Error	Test Error
1	1	6	0.4	6	6	4	32%	3400	11%	11%	7.3%	8.15%
2	5	6	2	6	12	4	45%	6400	12%	11%	7.4%	7.85%
3	4	6	2	6	12	4	42%	3400	12%	11%	7.5%	8.15%

Let the class conditional densities for the two classes be given by the Gaussian distributions $P(X|\omega_1) = N(m_1, \Sigma_1)$ and $P(X|\omega_2) = N(m_2, \Sigma_2)$, where:

$$m_1 = [2,2]^T, \quad m_2 = [4,4]^T,$$

$$\Sigma_1 = \begin{bmatrix} 1 & -0.25 \\ -0.25 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 1.5 & -0.25 \\ -0.25 & 1.5 \end{bmatrix}.$$

The following discriminant function is considered for this problem:

$$g(X)$$

$$= \left[mx_2 + x_1 - (m^2+1)\left(x_0 - \frac{b}{(1+m^2)^{1/2}}\right)\right]^2 \frac{1}{1+m^2}$$

$$- \left[x_1 - \left(x_0 + \frac{b}{(1+m^2)^{1/2}}\right)\right]^2$$

$$- \left[x_2 - m\left(x_0 + \frac{b}{(1+m^2)^{1/2}}\right)\right]^2 .$$

This function is a parabola described by three parameters, m, x_0 and b. The parameters of the optimal discriminant function for the above pattern classification problems are $m = 1.0, x_0 = 3.0$ and $a = 10.0$. For solving this problem, a team of three CALA are used, where each CALA is associated with one unknown parameter. The other parameters for the CALA used in this algorithm are the same as the parameters used in the previous section. For learning, 300 samples of pattern are chosen from each class, which are presented repeatedly to the team of CALA during the training process. Also, a test set consisting of 100 samples is generated from the two classes to find errors by the chosen discriminant function. The results of simulation for the proposed algorithm and the algorithm given in [8] are shown in Table 12. By carefully inspecting Table 12, it can be concluded that the proposed algorithm has smaller training and generalization errors.

CONCLUSIONS

In this paper, a random search algorithm for solving stochastic optimization problems has been studied. A class of problems has been considered where the only available information is noise-corrupted values of the function. In order to solve such problems, a new continuous action-set learning automaton was proposed and its convergence properties were studied. A strong convergence theorem for the proposed learning automaton was stated and proved. Finally, algorithms for two stochastic optimization problems were given and their behavior was studied through computer simulations. Computer simulations showed that the proposed method has a higher performance compared with existing continuous action-set learning automata based on optimization algorithms.

ACKNOWLEDGMENT

This research was, in part, supported by a grant from the Institute for Studies in Theoretical Physics and Mathematics (IPM) (No. CS1385-4-04), Tehran, Iran.

REFERENCES

1. Kushner, H.J. and Yin, G.G. "Stochastic approximation algorithms and applications", *Applications of Mathematics*, New York, Springer-Verlag (1997).
2. Narendra, K.S. and Thathachar, K.S. *Learning Automata: An Introduction*, Printice-Hall, New York, USA (1989).
3. Najim, K. and Pozyak, A.S., *Learning Automata: Theory and Applications*, Oxford, Pergamon press (1994).
4. Thathachar, M.A.L. and Sastry, P.S. "Varieties of learning automata: An overview", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, **32**, pp 711-722 (Dec. 2002).
5. Barto, A.G. and Anandan, P. "Pattern-recognizing stochastic learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-15**, pp 360-375 (May 1985).
6. Thathachar, M.A.L. and Satstry, P.S. "Learning optimal discriminant functions through a cooperative game of automata", *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-17**, pp 73-85 (Jan. 1987).

7. Najim, K. and Pozyak, A.S. "Multimodal searching technique based on learning automata with continuous input and changing number of actions", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, **26**, pp 666-673 (Aug. 1996).
8. Santharam, G., Sastry, P.S. and Thathachar, M.A.L. "Continuous action set learning automata for stochastic optimization", *Journal of Franklin Institute*, **331B**(5), pp 607-628 (1994).
9. Frost, G.P. "Stochastic optimization of vehicle suspension control systems via learning automata", Ph.D Thesis, Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, Leicestershire, LE81 3TU, UK (Oct. 1998).
10. Howell, M.N., Frost, G.P., Gordon, T.J. and Wu, Q.H. "Continuous action reinforcement learning applied to vehicle suspension control", *Mechatronics*, **7**(3), pp 263-276 (1997).
11. Gullapalli, V. "Reinforcement learning and its application on control", Ph.D Thesis, Department of Computer and Information Sciences, University of Massachusetts, Amherst, MA, USA (Feb. 1992).
12. Vasilakos, A. and Loukas, N.H. "ANASA-a stochastic reinforcement algorithm for real-valued neural computation", *IEEE Transactions on Neural Networks*, **7**, pp 83-842 (July 1996).
13. Rajaraman, K. and Sastry, P.S. "Stochastic optimization over continuous and discrete variables with applications to concept learning under noise", *IEEE Transactions on Systems, Man, and Cybernetics- Part A: Systems and Humans*, **29**, pp 542-553 (Nov. 1999).
14. Nedzelnitsky, O.V. and Narendra, K.S. "Nonstationary models of learning automata routing in data communication networks", *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-17**, pp 1004-1015 (Nov. 1987).
15. Obaidat, M.S., Papadimitriou, G.I., Pomportsis, A.S. and Laskaridis, H.S. "Learning automata-based bus arbitration for shared-medium ATM switches", *IEEE Transactions on Systems, Man, and Cybernetics- Part B: Cybernetics*, **32**, pp 815-820 (Dec. 2002).
16. Papadimitriou, G.I. Obaidat, S.M.S. and Pomportsis, A.S. "On the use of learning automata in the control of broadcast networks: A methodology", *IEEE Transactions on Systems, Man, and Cybernetics- Part B: Cybernetics*, **32**, pp 815-820 (Dec. 2002).
17. Oommen, B.J. and de St. Croix, E.V. "Graph partitioning using learning automata", *IEEE Transactions on Computers*, **45**, pp 195-208 (Feb. 1996).
18. Meybodi, M.R. and Beigy, H. "Solving stochastic shortest path problem using distributed learning automata", in *Proceedings of 6th Annual Iran Computer Society of Iran Computer Conference CSICC-2001, Isfahan, Iran*, pp 70-86 (Feb. 2001).
19. Beigy, H. and Meybodi, M.R. "Solving the graph isomorphism problem using learning automata", in *Proceedings of 5th Annual International Computer Society of Iran Computer Conference, CISCC-2000, Tehran, Iran*, pp 402-415 (Jan. 2000).
20. Oommen, B.J. and Roberts, T.D. "Continuous learning automata solutions to the capacity assignment problem", *IEEE Transactions on Computers*, **49**, pp 608-620 (June 2000).
21. Oommen, B.J. and Roberts, T.D. "Discretized learning automata solutions to the capacity assignment problem for prioritized networks", *IEEE Transactions on Systems, Man, and Cybernetics- Part B: Cybernetics*, **32**, pp 821-831 (Dec. 2002).
22. Meybodi, M.R. and Beigy, H. "Neural network engineering using learning automata: Determining of desired size of three layer feedforward neural networks", *Journal of Faculty of Engineering*, **34**, pp 1-26 (Mar. 2001).
23. Meybodi, M.R. and Beigy, H. "A note on learning automata based schemes for adaptation of BP parameters", *Journal of Neurocomputing*, **48**, pp 957-974 (Nov. 2002).
24. Meybodi, M.R. and Beigy, H. "New learning automata based algorithms for adaptation of backpropagation algorithm parameters", *International Journal of Neural Systems*, **12**, pp 45-68 (Feb. 2002).
25. Beigy, H. and Meybodi, M.R. "Backpropagation algorithm adaptation parameters using learning automata", *International Journal of Neural Systems*, **11**, pp 219-228 (June 2001).
26. Beigy, H. and Meybodi, M.R. "Adaptive uniform fractional channel algorithms", *Iranian Journal of Electrical and Computer Engineering*, **3**, pp 47-53 (Winter-Spring 2004).
27. Beigy, H. and Meybodi, M.R. "Call admission in cellular networks: A learning automata approach", *Springer-Verlag Lecture Notes in Computer Science*, **2510**, pp 450-457, Springer-Verlag (Oct. 2002).
28. Beigy, H. and Meybodi, M.R. "A learning automata based dynamic guard channel scheme", *Springer-Verlag Lecture Notes in Computer Science*, **2510**, Springer-Verlag, pp 643-650 (Oct. 2002).
29. Beigy, H. and Meybodi, M.R. "An adaptive uniform fractional guard channel algorithm: A learning automata approach", *Springer-Verlag Lecture Notes in Computer Science*, **2690**, Springer-Verlag, pp 405-409 (Mar. 2003).
30. Doob, J.L., *Stochastic Processes*, New York: John Wiley and Sons, USA (1953).