STOCHASTIC SEARCHING ON THE LINE AND ITS APPLICATIONS TO PARAMETER LEARNING IN NONLINEAR OPTIMIZATION*

B. John Oommen[1]
Professor, School of Computer Science
Carleton University
Ottawa, Canada : K1S 5B6.

ABSTRACT

We consider the problem of a learning mechanism (for example, a robot) locating a point on a line when it is interacting with an random environment which essentially informs it, possibly erroneously, which way it should move. In this paper we present a novel scheme by which the point can be learnt using some recently devised learning principles. The heart of the strategy involves discretizing the space and performing a controlled random walk on this space. The scheme is shown to be ε-optimal and to converge with probability 1. Although the problem is solved in its generality, its application in non-linear optimization has also been suggested. Typically, an optimization process involves working one's way toward the maximum (minimum) using the local information that is available. However, the crucial issue in these strategies is that of determining the parameter to be used in the optimization itself. If the parameter is too small the convergence is sluggish. On the other hand, if the parameter is too large, the system could erroneously converge or even oscillate. Our strategy can be used to determine the best parameter to be used in the optimization.

I. INTRODUCTION

Consider the problem of a robot (algorithm, learning mechanism) moving around on the real line attempting to locate a particular point. To assist the mechanism it communicates with an Environment ("Oracle") directing it with information regarding the direction in which it should go. If the Environment is deterministic the problem is the "Deterministic Point Location Problem" and that has been studied rather thoroughly [1]. Strictly speaking, this problem involves searching with uncertainty, because the destination is unknown, and all that is known is the robot's direction of motion. In its pioneering version [1] the problem was presented in the setting that the Environment ("Oracle") could charge the robot a cost proportional to the distance to the destination. The question of having multiple communicating robots locate a point on the line has also been studied [1,2].

In this paper we consider a generalized version of this problem. We consider the scenario when a robot (or more generally, a learning mechanism) is moving within an interval. It is attempting to locate a

* Partially supported by the Natural Sciences and Engineering Research Council of Canada. A preliminary version of some of the results in this paper can be found in the Proceedings of the 1993 International Symposium on Nonlinear Theory and its Applications, Hawaii, December 1993.
[1]Senior Member, IEEE.

point in this interval. However, rather than receive deterministic responses as to where it should go, it is, at every time interval, given a stochastic (i.e., possibly erroneous) response. Thus when it should really be moving to the "right" it may be advised to move to the "left" and vice versa. This problem, referred to as the "Stochastic Point Location Problem", is the focus of this paper, and we believe that our results are the first results on this problem.

In this paper we present a novel scheme to solve the latter problem. The scheme uses some recently devised learning principles which resort to discretizing the space [6,7,9,11,18] and performing a controlled random walk on this space. The learning scheme is shown to be ε-optimal, and thus we can reach the point with an arbitrarily small error. It also converges with probability 1.

Apart from the problem being of importance in its own right, it also has potential applications in solving optimization problems. The goal of the latter problems is to attempt to accomplish a given task with the minimum cost or with the maximum benefits. If the underlying cost function (or benefit function) is known, the problem at hand is usually one of minimizing (maximizing) this function. Furthermore, in many applications the optimization is constrained, the constraints being based on the restrictions imposed by the problem on the arguments or variables involved.

In many optimization solutions, the algorithm works its way from its "current" solution to the optimal one based on currently available information. The crucial question is one of determining the parameter which the optimization algorithm should use. In a typical hill-climbing algorithm, if the parameter is too small the convergence is sluggish. On the other hand, if the parameter is too large, the system could erroneously converge or even oscillate. In many cases the parameter of the scheme is related to the second derivative of the criterion function, which results in a technique which is analogous to a "Newton's" root solving scheme. But the disadvantages of the latter are well known : If the starting point of the algorithm is not well chosen, the scheme can diverge. Furthermore, if the second derivative is small, the scheme is ill-defined. Finally, such a scheme requires the additional computation involved in evaluating the (matrix of) second derivatives [15,16,20].

A couple of examples can serve to illustrate the importance of the choice of the parameter in typical optimization problems. In image processing, it is customary that a polygon is approximated, for example, using its minimum perimeter polygon [4,14]. If the original polygon has N points, it can be shown that the latter can be obtained by solving a sequence of quartic polynomial equations. This is because the problem reduces to an optimization involving quadratic programming because the constraints are that the approximated points should lie within circular disks centered at the original points. In this case, if the parameter chosen in the optimization algorithm is too small, the convergence is extremely sluggish. But if it is even a trivial too large, the algorithm typically goes "out of control" to yield infeasible solutions, because the problem is extremely ill-conditioned.

Another more interesting example is the optimization that the back propagation neural network achieves [13,17,20]. Given a particular input, the network uses its "current" set of weights to compute the corresponding output. The obtained output is compared to the expected output and the network weights are consequently modified so as to minimize the expected resultant error. Here too, since we are dealing with a multi-variable minimization, the parameter of choice is crucial. Indeed, much of the research that has gone into enhancing the convergence rate of the algorithm has involved updating the weights without explicitly evaluating second derivatives.

Various other applications of optimization can be mentioned. But in the interest of brevity we shall merely refer the reader to an excellent book by Rao [16] for a superb treatment of the theory and applications of optimization and the need for wisely choosing the parameters of the algorithms.

To overcome this drawback we suggest that our strategy to solve the stochastic point location problem can be used to learn the best parameter to be used in any algorithm. The scheme requires no computation of any additional derivatives. Thus, if there is a best parameter and a (possible stochastic) method of determining whether we are moving towards or away from this parameter, then a learning strategy can be employed to converge to a value arbitrarily close to this best parameter.

The heart of the strategy involves discretizing the parameter space and performing a controlled random walk on this space. The beauty of a discrete Learning Algorithm (LA) is that it does not ignore the limitations of practical implementations -- on the contrary this limitation is used to its advantage. Traditional LAs (also referred to as variable structure stochastic automata) evolved from fixed structure stochastic automata as an attempt to simplify the analysis of the automata's properties [5,7]. However, the former algorithms have a limitation. Implicit in their definition is the fact that the probability of choosing an action can be any real number in the interval [0, 1]. Rendering this probability space discrete is a general approach for improving the learning properties of the algorithms [6,7,9,11,18]; this is implemented by restricting the probability of choosing an action to only finitely many values from the interval [0, 1]. In these algorithms, the probability changes are made in jumps and not continuously, and thus the speed of the learning process is increased. Philosophically, discrete machines represent a hybrid of the fixed and variable structured families.

The existing results about discretized automata are found in [6,7,9-12,18]. Indeed, the fastest reported learning automata are the discretized pursuit and estimator algorithms [6,7].

We can summarize the advantages of using discretized learning algorithms (LA) as follows. Discrete LAs increase the rate of convergence of the learning process and eliminate the assumption that the random number generator can generate real numbers with arbitrary precision. Once the optimal action has been determined, the discrete automaton increases the probability of choosing the correct action directly, rather than approach the value of unity asymptotically and thus avoid the pitfalls caused by inherent round-off errors. Finally, and far from being unimportant are the considerations of implementation and representation. Discrete versions lead, quite naturally, to the use of integers for keeping track of how many multiples of $^1/_N$ the action probabilities are. While the above consideration frequently increases the rate of convergence measured in terms of the number of iterations, a discrete algorithm also has the benefit of reducing the time measured in terms of the clock cycles that a microprocessor would take to do **each** iteration of the task. It also reduces the amount of memory needed. Typically addition is quicker than multiplication, and the amount of memory used for a floating point number is usually more than that required for an integer. Thus, in terms of both time and space, discrete LAs seem to be superior.

II. THE STOCHASTIC POINT LOCATION PROBLEM

We assume that there is a learning mechanism whose task is to determine the optimal value of some variable (or parameter), λ. We assume that there is an optimal choice for λ -- an unknown value, say λ^*. The question which we study here is one of learning λ^*. Although the mechanism does not know the value of λ^* we assume that it has responses from an intelligent "environment" ⦅ ("oracle") which is capable of informing it whether the current value of λ is too small or too big. To render the problem both meaningful and distinct from its deterministic version, we would like to emphasize that the response from this environment is assumed "faulty". Thus, ⦅ may tell us to increase λ when it should be decreased, and vice versa. However, to render the problem tangible we assume that the probability of receiving an intelligent response is $p > 0.5$.

The quantity 'p' reflects on the "effectiveness" of the environment, ⦅. Thus, whenever the current $\lambda < \lambda^*$, the environment correctly suggests that we increase λ with probability p. It simultaneously could have incorrectly recommended that we decrease λ with probability (1-p). Similarly, whenever $\lambda > \lambda^*$, the environment tells us to decrease λ with probability p, and to increase it with probability (1-p).

We shall assume that λ is any number in the interval [0,1]. The question of generalizing this will be considered later. The crucial issue that we have to address is that of determining how to change our guess of λ^*. We shall attempt to do this in a discretized manner by subdividing the unit interval into N steps $\{0, {}^{1}/_{N}, {}^{2}/_{N}, ..., {}^{(N-1)}/_{N}, 1\}$, where N is the resolution of the learning scheme. A larger value of N will ultimately imply a more accurate convergence to the unknown λ^*.

The scheme which attempts to learn λ^* is as below. Let $\lambda(n)$ be the value at time step 'n'. Then,

$\lambda(n+1) := \lambda(n) + 1/N$ If ⦅ suggests increasing λ and $0 \quad \lambda(n) < 1$, (1)

$\lambda(n+1) := \lambda(n) - 1/N$ If ⦅ suggests decreasing λ and $0 < \lambda(n) \quad 1$. (2)

At the end states the scheme obeys :

$\lambda(n+1) := \lambda(n)$ If $\lambda(n) = 1$ and ⦅ suggests increasing λ (3)

$\lambda(n+1) := \lambda(n)$ If $\lambda(n) = 0$ and ⦅ suggests decreasing λ. (4)

Notice that although the above rules are deterministic, because the "environment" is assumed faulty, the state transitions are stochastic. We now prove our main first result concerning the above scheme.

Theorem I

The parameter learning algorithm specified by (1)-(4) is asymptotically optimal.

Proof :

Our intention is to prove that as N is increased indefinitely,

$$\underset{N\varnothing}{\text{Lim}} \ \underset{n\varnothing}{\text{Lim}} \ E\left[\lambda(n)\right] \varnothing \lambda^*.$$

We shall prove this by analyzing the properties of the underlying Markov chain which is specified by the rules (1)-(4). The states of the chain are the integers {0,1,2,..., N} corresponding to the values $\{0, {}^{1}/_{N}, {}^{2}/_{N}, ..., {}^{(N-1)}/_{N}, 1\}$ respectively.

If $\phi(n)$ is the state at 'n', from (1)-(4) we can write the transition probabilities of the chain as :

$\phi(n+1) := \phi(n) + 1$ If ⦅ suggests increasing λ and $0 < \phi(n) < N-1$.

$\phi(n+1) := \phi(n) - 1$ If ⦅ suggests decreasing λ and $1 < \phi(n) < N$.

At the end states,

$\phi(n+1) := \phi(n)$ If ⦅ suggests increasing λ and $\phi(n) = N$.

$\phi(n+1) := \phi(n)$ If ⦅ suggests decreasing λ and $\phi(n) = 0$.

Let Z be the index for which

$Z/N < \lambda^* < (Z+1)/N$.

Then, λ^* lies between the values represented by the states Z and (Z+1).

Let q = 1-p. The above rule obeys the Markov chain with a transition matrix T, where :

$T_{i,i+1}$ = p if 0 i Z (5)

= q if Z < i N-1. (6)

$T_{i,i-1}$ = q if 1 < i Z (7)

= p if Z < i N. (8)

The transitions for the boundary states specified by (3) and (4) are self loops and are given as:

$T_{0,0}$ = q (9)

$T_{N,N}$ = q (10)

We shall now compute π_i, the stationary (or equilibrium) probability of the chain being in state i. Clearly T represents a single closed communicating class whose periodicity is unity. The chain is ergodic, and the limiting probability vector is given by the eigenvector of T^T corresponding to the eigenvalue unity. Let this vector be $\Pi = [\pi_0, ..., \pi_N]$. Then, Π satisfies :

$$\begin{bmatrix} q & p & 0 & . & . & . & 0 & 0 & . & . & 0 & 0 \\ q & 0 & p & . & . & . & 0 & 0 & . & . & 0 & 0 \\ 0 & q & 0 & p & . & . & 0 & 0 & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . & . & . & . & . \\ & & & & & & & & & & & \\ . & . & . & . & . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & p & . & . & . & . & . \\ 0 & 0 & 0 & . & . & q & 0 & p & . & . & . & . \\ 0 & 0 & 0 & . & . & 0 & p & 0 & q & . & 0 & 0 \\ 0 & 0 & 0 & . & . & . & 0 & p & 0 & . & 0 & 0 \\ 0 & 0 & 0 & . & . & . & . & . & . & . & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & 0 & q \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p & q \end{bmatrix}^T \begin{bmatrix} \Pi_0 \\ \Pi_1 \\ \Pi_2 \\ . \\ . \\ . \\ \Pi_{Z-1} \\ \Pi_Z \\ \Pi_{Z+1} \\ . \\ \Pi_{N-1} \\ \Pi_N \end{bmatrix} = \begin{bmatrix} \Pi_0 \\ \Pi_1 \\ \Pi_2 \\ . \\ . \\ . \\ \Pi_{Z-1} \\ \Pi_Z \\ \Pi_{Z+1} \\ . \\ \Pi_{N-1} \\ \Pi_N \end{bmatrix} \quad (11)$$

The details of solving (11) are quite cumbersome and follow as a consequence of the doubly stochastic properties of T for *almost* all the interior states of the chain. We claim that if e = p/q, (note that e > 1) these obey :

$\pi_i = e.\pi_{i-1}$ whenever i   Z. (12)

$\pi_i = \pi_{i-1} / e$ whenever $i > Z+1$, and, (13)

$\pi_{Z+1} = \pi_Z$. (14)

We prove (12)-(14) by induction. For the basis step we expand the first row of (11) to yield :

$q\pi_0 + q\pi_1 = \pi_0$

whence it follows that $\pi_1 = e.\pi_0$.

Since we have a second order "difference" equation relating the π_k's, we continue the basis step by expanding the second row of (11) to yield :

$p\pi_0 + q\pi_2 = \pi_1$

Utilizing the fact that $\pi_1 = e.\pi_0$ we obtain :

$p\pi_0 + q\pi_2 = e.\pi_0$

which simplifies to yield $\pi_2 = e^2.\pi_0$.

For the inductive step for $i < Z$ we now expand the $k^{th.}$ row of (11) to yield :

$p\pi_{k-1} + q\pi_{k+1} = \pi_k$ (15)

Using the premise of the induction that :

$\pi_{k-1} = e^{k-1}.\pi_0$ and $\pi_k = e^k.\pi_0$

(15) reduces to :

$pe^{k-1}.\pi_0 + q\pi_{k+1} = e^k.\pi_0$

whence it can be seen that π_{k+1} has to satisfy $\pi_{k+1} = e^{k+1}.\pi_0$.

We now consider the case for π_Z and π_{Z+1}. The former obeys :

$$p\pi_{Z-1} + p\pi_{Z+1} = \pi_Z \qquad (16)$$

We shall first assume that $\pi_Z = \pi_{Z+1}$ in a straightforward substitution, and show that it satisfies (11). Since the solution to (11) is the *unique* eigenvector of the eigenvalue unity, this assumption therefore has to be true. Setting π_{Z+1} to be equal to π_Z, (16) yields the value of π_Z and π_{Z+1} to be :

$$\pi_Z \quad = e^Z.\pi_0$$

$$\pi_{Z+1} = e^Z.\pi_0$$

We now have to solve for π_k for the rest of the states of the chain (i.e., for $Z+2 < k < N-1$). First of all observe that for π_{Z+2} the equation is :

$$p\pi_Z + p\pi_{Z+2} = \pi_{Z+1}$$

and since $\pi_Z = \pi_{Z+1} = e^Z.\pi_0$, we obtain :

$$\pi_{Z+2} = e^{Z-1}.\pi_0.$$

Note that subsequent to this, the difference equation that is obeyed is a mirror reflection of the one obeyed prior to the state Z, and we can therefore anticipate that the values of π_k will diminish geometrically. We shall now show that this is indeed the case. For all k satisfying $Z+2 < k < N-1$ the equation obtained by expanding the corresponding row of (11) is :

$$q\pi_{k-1} + p\pi_{k+1} = \pi_k \qquad (17)$$

Since it is true for π_{Z+1} and π_{Z+2} that $\pi_{Z+2} = \pi_{Z+1}/e$ we can use this as the premise of the induction for $Z+2 < k < N-1$ and obtain :

$$q\pi_{Z+1} + p\pi_{Z+3} = \pi_{Z+2}$$

whence $\pi_{Z+3} = \pi_{Z+2}/e$.

Arguing in a similar way it is easy to see that for all $Z+1 \quad k < N-1$,

$$\pi_{k+2} = \pi_{k+1}/e,$$

or rather that for all $Z+1 \quad k < N-1$

$$\pi_k = e^{Z-(k-(Z+1))}.\pi_0, \text{ which is the same as}$$

$$\pi_k = e^{2Z-k+1}.\pi_0.$$

The final value for π_N follows since *its* difference equation is :

$$q\pi_{N-1} + q\pi_N = \pi_N$$

whence it is easily seen that $\pi_N = \pi_{N-1}/e$.

To conclude the proof of the theorem we have to now consider the equilibrium value of $E[\lambda(n)]$ for any finite N, and then take this limit as $N \emptyset$. We shall consider the limiting arguments in three cases. First of all, when the limit of the ratio Z/N is close to zero, the value of π_i quickly reaches its maximum at π_Z and then falls away geometrically. Thus, as $N \emptyset$, most of the probability mass will be concentrated in an arbitrarily small interval centered around Z. Similarly when the limit of the ratio Z/N is close to unity, the value of π_i increases geometrically till it reaches its maximum at π_Z and then falls away rapidly. Again, as N ∅   most of the probability mass will be concentrated in an arbitrarily small interval centered around Z.

The complicated case is the case when the limit of the ratio Z/N is bounded away from the boundaries zero and unity. In this case we have two geometric series -- the first of which increases from π_0 to π_Z, and the second decreases from π_{Z+1} to π_N. Here, we shall consider the two series separately. Since e > 1, the mass associated with the states increases geometrically with the state indices till the state Z. It then decreases geometrically, with the same factor, e, from state Z+1 to N. Since the limit of the ratio Z/N is bounded away from both zero and unity, for the first series of masses from π_0 to π_Z, since e^i increases exponentially and we are speaking about the mean of an increasing geometric progression, most of the mass will be concentrated among an arbitrarily small number of states close to Z. Analogously, in this case for the second series of masses from π_{Z+1} to π_N, since $1/(e^i)$ decreases exponentially and since we are speaking about the mean of *this* geometric progression, the mass will be concentrated among an arbitrarily small number of states close to Z+1. Thus, as N∅ , the mean of $E[\lambda(\ )]$ will lie in an arbitrarily small interval centered about Z/N and (Z+1)/N, and will thus be arbitrarily close to λ^*. Hence the theorem.

In the literature there is another common definition for ε-optimality. If p_b is the probability of a learning machine choosing the best action, the machine is said to be ε-optimal if for all ε > 0,

$$\underset{n\varnothing}{\text{Lim}} \ \inf E[p_b(n)] > 1 - \varepsilon.$$

Intuitively this definition requires that the probability of picking the best action is arbitrarily close to unity. Although the relationship between this definition and the one used above has not been worked out for general learning automata, in our present study we can prove a stronger result than Theorem I that $\lambda(\ )$ converges to λ^* not only in the expected sense, but also with probability one.

Theorem II

The learning algorithm specified by (1)-(4) yields $\lambda(\ )$ which converges to λ^* with prob. 1.

Proof :

It is well known that if ξ is a random variable in the interval [0,1], then

$\xi \varnothing 0$ with prob. 1 if $E[\xi] \varnothing 0$, and,

$\xi \varnothing 1$ with prob. 1 if $E[\xi] \varnothing 1$.

The result is therefore obvious if λ^* is in the neighborhood of zero or unity.

The interesting case is when λ^* is bounded away from the end points of the unit interval. We define the indicator function I_λ as :

$$\begin{aligned} I_\lambda \quad &:= \quad 1 \quad \text{if } \lambda < \lambda^* \\ &:= \quad 0 \quad \text{if } \lambda \quad \lambda^*. \end{aligned}$$

We now utilize the final limiting arguments of Theorem I. As argued above, in this case we have two geometric series -- the first of which increases from π_0 to π_Z, and this would occur with $q_1(\ )$, the probability associated with the event $I_{\lambda(\ )} = 1$. The second series decreases from π_{Z+1} to π_N, and would occur with $q_0(\ )$, the probability associated with the event $I_{\lambda(\ )} = 0$. As before, we shall consider the two series separately.

For the first series, since e > 1, the mass associated with the states increases geometrically with the state indices till the state Z. Since e^i increases exponentially, we know that the mean of the increasing geometric progression yields an expected value arbitrarily close to the quantity Z/N. Thus, as N tends to infinity, $E[\lambda(\ ) \mid I_{\lambda(\ )} = 1] \varnothing Z/N$, and hence,

$$\lambda(\ ) \mid I_{\lambda(\ )} = 1 \ \varnothing\ Z/N \qquad \text{with prob. 1.} \tag{18}$$

For the second series, since e > 1, the mass associated with the states decreases geometrically with the state indices starting from the state (Z+1) till the state N. Again this exponential decrease dictated by e^i implies that the expected value of the decreasing geometric progression is arbitrarily close to the quantity (Z+1)/N. Thus, as $N \varnothing$, $E[\lambda(\ ) \mid I_{\lambda(\ )} = 0] \varnothing (Z+1)/N$, and hence,

$$\lambda(\ ) \mid I_{\lambda(\ )} = 0 \ \varnothing\ (Z+1)/N \qquad \text{with prob. 1.} \tag{19}$$

As a consequence of (18) and (19)

$$\Pr\left[Z/N < \lambda(\ ) \quad (Z+1)/N\ \right] \varnothing 1.$$

The result follows since for all N,

$$Z/N < \lambda^* \quad (Z+1)/N,$$

and hence[2] there exists a $\delta \varnothing 0$ such that,

$$\lim_{N\varnothing} \Pr\left[\lambda^* - \delta < \lambda(\ ) \quad \lambda^* + \delta\right] \varnothing 1.$$

Hence the theorem.

II.1 Example

A simple example will help clarify matters. Suppose we partition the interval [0,1] into eight intervals as $\{0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1.0\}$. Also let λ^* be 0.65. Then,

(i) All transitions for values of λ in {0, 0.125, 0.25, 0.375, 0.5, 0.625} will be increased with a probability of p and decreased with a probability of q, and,

(ii) All transitions for values of λ in {0.75, 0.875, 1.0} will be decreased with a probability of p and increased with a probability of q.

In this case, it can be seen that if e := p/q, then, if K is a normalizing constant

$$\pi_0 := K \ ; \qquad \pi_i := K.e^i, \quad \text{for } 0 \quad i \quad 5,$$
$$\pi_6 := K.e^5 \qquad \pi_i := K.e^{11-i} \quad \text{for } 7 \quad i \quad 8.$$

π_i increases geometrically from π_1 to π_5 and decreases geometrically from π_6 to π_8 with $\pi_5 = \pi_6$.

As $N \varnothing$, since e > 1, all the probability mass gets concentrated on states arbitrarily close to states neighboring λ^*. Indeed, in the limit $\lambda(\ )$ converges to λ^* both in mean and with prob. 1

[2]Note that the following limit operation is done on the resolution parameter 'N'. As opposed to this, the parenthesized parameter for λ, given by 'n', represents time.

III. STOCHASTIC POINT LOCATION AND OPTIMIZATION

Although the problem which we have solved is interesting in its own right and a generalization of its deterministic version, it also has powerful applications in non-linear optimization. As mentioned earlier, typically the crucial issue in these strategies is that of determining the parameter to be used in the optimization itself. If the parameter is too small the convergence is sluggish, and, if the parameter is too large, the system could erroneously converge or even oscillate. The learning strategy presented here can be used to determine the best parameter to be used in the optimization.

Throughout the previous section we had assumed that we are dealing with the problem of learning the optimal parameter when we were able to communicate with a relatively intelligent "environment" (capable of stochastically informing us whether the current value of λ should be decreased or increased. We shall now consider the question of utilizing our solution in optimization by simulating the environment from the actual measurements of the criterion function.

We first consider the question of relaxing the assumption that λ has to be in the interval [0,1]. With no loss of generality, if the bounds on the parameter are known, this can clearly be achieved by a simple normalizing function, . Thus, if it is known that the parameter to be used, μ, should always be between μ_{min} and μ_{max}, a simple linear transformation given by :

$$\lambda = (\mu - \mu_{min})/(\mu_{max} - \mu_{min})$$

would render the normalized parameter λ to be in the unit interval [0,1]. However, in the case of functions such as those used in neural networks [16,17,20] it is well known that the range of the parameter can vary from 10^{-3} to 10^{3}. In such a case the linear partitioning of the space is both inconvenient and cumbersome. In such cases we recommend the use of a monotonic *one-to-one* mapping from λ to the actual parameter space by a function such as :

$$\lambda := A. \operatorname{Log}_b \mu$$

for some convenient constant, b. Since λ converges to a value arbitrarily close to λ^*, since the function is both monotonic and one-to-one, μ will converge arbitrarily close to its ideal value, μ^*.

To delve into the optimization problem itself, suppose that the optimization involves minimizing a criterion function Ξ, where the variable of optimization is x. Typically, we are looking for the place when the partial derivative $\delta\Xi/\delta x$ is zero. If Ξ is positive, the simple linear rule would thus move 'x' in the direction of the solution. Generally speaking, the second derivative information can be further utilized to specify how much the move should be. As a first approximation for the second derivative we suggest a scheme analogous to the RPROP network [17]. RPROP (for Resilient Propagation) is an adaptive learning algorithm that considers the local topology of the criterion function based on the Manhattan-Learning Rule described by Sutton and referred to in [17]. Here Δx would be :

$$\begin{aligned} \Delta x \quad &= \quad -\Delta_0 \qquad \text{If} \quad \delta\Xi/\delta x > 0 \\ &= \quad \Delta_0 \qquad \text{If} \quad \delta\Xi/\delta x < 0 \end{aligned}$$

$= \quad 0 \qquad$ Otherwise,

where Δ_0, the size of the increment is a problem-dependent constant.

Due to its simplicity, the above scheme for modifying the variable is very crude, and so the method does not generally work satisfactorily on difficult nonlinear problems. The fundamental idea for the improvement suggested by the RPROP algorithm is to "squeeze out" more information about the topology of the criterion function. In this case, in the setting of neural networks, each weight w_{ij} has its own increment value Δ_{ij}, which evolves with the learning process according to the local perception of the criterion function. The RPROP learning rule is as follows :

$$\begin{aligned}
\Delta_{ij}(t) \quad &= \quad -\Delta_{ij}(t-1)\cdot\eta^{+} && \text{If} \quad \frac{\delta\Xi(t-1)}{\delta w_{ij}} \cdot \frac{\delta\Xi(t-1)}{\delta w_{ij}} > 0, \\
&= \quad -\Delta_{ij}(t-1)\cdot\eta^{-} && \text{If} \quad \frac{\delta\Xi(t-1)}{\delta w_{ij}} \cdot \frac{\delta\Xi(t-1)}{\delta w_{ij}} < 0, \\
&= \quad \Delta_{ij}(t-1) && \text{Otherwise,}
\end{aligned}$$

where η^{-} and η^{+} are the parameters of the scheme satisfying $0 < \eta^{-} < 1 < \eta^{+}$. Typical values for η^{-} and $\eta^{+}$are 0.5 and 1.2 [17].

In this case the increments are influenced by the behaviour of the **sign** of two succeeding derivatives and not by the magnitudes of the derivatives themselves. Whenever the partial derivative of the weight w_{ij} changes its sign indicating that the last update was too big and the algorithm has jumped over a local minimum, the increment is decreased by η^{-}. If the derivative retains its sign, the increment is slightly increased in order to accelerate convergence in shallow regions.

This is exactly the philosophy that we recommend for designing (. If the partial derivative changes sign, (will suggest that the value of λ be decremented. Otherwise, (suggests that it be incremented. Obviously, this essentially implies that whenever the partial derivative retains the same sign we are, in all likelihood, still on the same side of the optimum. We could thus venture to take a controlled, marginally larger step, where the magnitude of the increase in λ (not in the optimization variable itself) is controlled by the resolution parameter, N. But when the partial derivative changes its sign we have possibly overstepped the optimum, and this is undone by recommending that λ be decremented. Thus the change in the increments depends only on the sign of the partial derivative without reference to its magnitude. What we are thus effectively attempting to achieve is to "simulate" a mechanism which mimics the Newton's Rule without actually evaluating the second derivatives.

We are currently investigating how the scheme presented here can be utilized to catalyze the convergence of various well-defined optimization problems. We are also studying how it can be used to generalize the concepts introduced by Riedmiller *et. al.* for hastening the convergence of neural networks [17].

IV. EXPERIMENTAL RESULTS

The parameter learning mechanism described in this paper was experimentally evaluated to verify the validity of our analytic results and to examine its rate of convergence.

The experiments were primarily of two types. In the first type of experiments, in order to obtain robust results, the value for $E[\lambda(\ )]$ was evaluated by computing the closed-form expression of the asymptotic distribution and subsequently evaluating its mean terminal value. In the second type of experiments, in order to obtain an understanding as to how the scheme converged with time, various simulations were conducted to evaluate the performance of the algorithm under a variety of constraints. In each simulation, a **hundred** parallel experiments were conducted so that an accurate ensemble average of the results could be obtained. In each case, the value of the parameter λ^* was assumed unknown to the learning mechanism, and the simulation was carried out for various values of p, the probability of the environment responding correctly. The average of the asymptotic value of λ was recorded for each of the parallel experiments.

The experimental results obtained are truly conclusive. Although hundreds of experiments have been conducted, in the interest of brevity we shall merely report some results which highlight the salient features of our scheme. In Table I we have recorded the true value of $E[\lambda(\ )]$ for various values of p and for various resolutions when the value of λ^* is 0.9123. In every case the convergence of $E[\lambda(\ )]$ is remarkable. For example, when p is 0.7, the value of $E[\lambda(\ )]$ is as high as 0.747 when N is as small as 4. It increases to 0.886 when N is 16 and comes within 0.5 **percent** of the true value before a resolution of N=32. The results are, of course, more "spectacular" for larger values of p. Thus, for example, when p is 0.85, the value of $E[\lambda(\ )]$ is as high as 0.834 when N is as small as 4. It comes to within 0.5 percent of the true value before a resolution of N=8. The power of the scheme is obvious !!

******************* Insert Table I *******************

A plot of the asymptotic value of $E[\lambda(\ )]$ with N for various values of p is given in Figure I. A similar plot of the asymptotic value of $E[\lambda(\ )]$ as a function of p is given in Figure II for various resolutions, N.

******************* Insert Figures I & II *******************

We now report the results of the second set of experiments in which we have tried to catalogue the convergence of $E[\lambda(n)]$ with time, 'n'. As mentioned earlier, in each simulation, the ensemble average was obtained by performing one hundred parallel experiments. Although in the actual simulations, the resolution of the mechanism, N, was also maintained as a variable, in the interest of brevity, we shall merely report the results for some specific values of N. The convergence of the scheme is remarkable even for relatively small values of N. For example, when p=0.7 and N=64, the experimental value of $E[\lambda(n)]$ increases from its initial value of 0.5 (at n=0) to 0.591 when n=16. Before the 128th. iteration the ensemble average was 0.910, which is less than 0.4 percent of the true value of λ^*. As can be expected, this convergence is even more rapid for larger values of p. To demonstrate this we report the results obtained for the case when p=0.85 and N=64. In this case, the value of $E[\lambda(n)]$ increases from its initial value of 0.5 to 0.592 when n=8. Before the 64th. iteration the ensemble average is 0.908 -- again, a value less than 0.5 percent of λ^*. A plot of the estimate of $E[\lambda(n)]$ with n for resolutions of N=16 and 64 are given in Figures III and IV respectively for various values of p. Observe the power of the algorithm !!

****************** Insert Figures III & IV ******************

V. CONCLUSIONS

In this paper we have considered the problem of a learning mechanism locating a point on a line when it is interacting with an random environment which essentially informs it, possibly erroneously, which way it should move. We have presented a novel scheme by which the point can be learnt using learning principles which involve discretizing the space and performing a controlled random walk on this space. The learning scheme converges ε-optimally and with prob. 1.. Although the problem is solved in its generality, its application in non-linear optimization has also been suggested. Traditionally, an optimization process involves working one's way toward the maximum (minimum) using the local information that is available. However, the crucial issue in these strategies is that of determining the parameter to be used in the optimization itself. If the parameter is too small the convergence is sluggish. On the other hand, if the parameter is too large, the system could erroneously converge or even oscillate. The learning strategy presented here can be utilized to determine the best parameter to be used in the optimization. Since the learning scheme is ε-optimal, if there is a "best" parameter which can achieve the optimization, our learning strategy can be made to converge to a value arbitrarily close to this parameter.

We are currently investigating how our scheme can be utilized to catalyze the convergence of various optimization problems including those involving neural networks as described in [17]. Another open problem involves studying how the discrete jumps made by the scheme can be of unequal sizes as opposed to the strategy proposed here. We believe that "stochastic-approximation" type of successively diminishing discrete jumps could lead to much faster schemes. Although such schemes can easily be described they would, typically, be very hard to analyze. Finally, the problem of analyzing the scenario when the value of 'p' is time/space varying remains open.

Acknowledgements : I am grateful to Ricardo Baeza-Yates for introducing me to this extremely interesting problem in Santiago, in October 1992. I am also grateful to Martin Riedmiller who introduced me to the RPROP algorithm and its variants. We are finally grateful to two anonymous referees who painstakingly read the paper and suggested various modifications.

REFERENCES

[1] Baeza-Yates, R. A., Culberson, J. C. and Rawlins, G. J. E., "Searching with uncertainty", *Proceedings of the Scandinavian Workshop on Algorithms and Theory*, SWAT 88, LNCS 318, 1988, pp. 176-189.

[2] Baeza-Yates, R. A. and Schott, R., "Parallel Searching in the Plane", *Proceedings of the 1992 International Conference of the Chilean Computer Society*, IC-SCCC 92, 1992, pp. 269-279.

[3] Karlin, S., and Taylor, H. M., *A First Course on Stochastic Processes* , 2nd. ed.,Academic Press , 1974.

[4] Kashyap, R.L., and Oommen, B.J., "Scale Preserving Smoothing of Polygons", *IEEE Trans. on Pat. Anal. and Mach. Intel.*, Nov. 1983, pp. 667-671.

[5] Lakshmivarahan, S., *Learning Algorithms Theory and Applications*, Springer-Verlag, 1981.

[6] Lanctôt, J. K.,*Discrete Estimator Algorithms: A Mathematical Model of Computer Learning*, M.Sc. Thesis, Department of Mathematics and Statistics, Carleton University, Ottawa, Canada, 1989.

[7] Lanctôt, J.K. and Oommen, B.J., "Discretized Estimator Learning Automata", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-22, November/December 1992, pp. 1473-1483.

[8] Narendra, K.S., and Thathachar, M.A.L., *Learning Automata*, Prentice-Hall, 1989.

[9] Oommen, B.J., and Hansen, E.R., "The Asymptotic Optimality of Discretized Linear Reward-Inaction Learning Automata", *IEEE Trans. on Syst. Man and Cybern.*, May/June 1984, pp.542-545.

[10] Oommen, B.J., and Christensen, J.P.R., "Epsilon-Optimal Discretized Linear Reward-Penalty Learning Automata", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-18, May/June 1988, pp. 451-458.

[11] Oommen, B.J., and Lanctôt, J.K., "Discretized Pursuit Linear Reward-Inaction Learning Automata", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-20, July/August 1990, pp. 431-438.

[12] Oommen, B.J., "Absorbing and Ergodic Discretized Two-Action Learning Automata", *IEEE Trans. on Syst. Man and Cybern.*, Vol. SMC-16, 1986, pp.282-296.

[13] Pao, Y.-H., *Adaptive Pattern Recognition and Neural Networks*, Addison Wesley, Reading, Mass., 1989.

[14] Pavlidis, T., *Structural Pattern Recognition*, Springer-Verlag, New York, 1977.

[15] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes : The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986.

[16] Rao, S.S., *Optimization :Theory and Applications*, John Wiley, New Delhi 2nd. Ed., 1984.

[17] Riedmiller, M. and Braun, H., "RPROP -- A Fast Adaptive Learning Algorithm", *Proceedings of the 1992 International Symposium on Computer and Information Sciences*, Antalya, Turkey, November 1992, pp.279-285.

[18] Thathachar, M.A.L., and Oommen, B.J., Discretized Reward-Inaction Learning Automata, *Journal of Cybernetics and Information Sciences*, Spring 1979, pp. 24-29.

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

[20] Wasserman, P. D., *Neural Computing : Theory and Practice*, Van Nostrand Reinhold, New York, 1989.