Parameter Learning from Stochastic Teachers and Stochastic Compulsive Liars

Govindachari Raghunath RAGHU@SCS.CARLETON.CA
John Oommen OOMMEN@SCS.CARLETON.CA
School of Computer Science, Carleton University
Ottawa, ON K1S-5B6, Canada

Benjamin Kuipers KUIPERS@CS.UTEXAS.EDU
Dept. Computer Sciences, University of Texas
Austin, Texas Texas 78712 USA

Abstract

We consider the problem of a learning mechanism (robot, or algorithm) that learns a parameter while interacting with either a *stochastic teacher* or a *stochastic compulsive liar*. The problem is modeled as follows: the learning mechanism is trying to locate an unknown point on a real interval by interacting with a stochastic environment through a series of estimates. For each estimate the environment (teacher) essentially informs the mechanism, possibly erroneously, which way it should move to reach the point. Thus, there is a non-zero fixed (i.e., stationary) probability that the feedback from the environment is erroneous. When the probability of correct response is $p > 0.5$, the environment is said to be *Informative* and we have the case of learning from a *stochastic teacher*. When this probability is $p < 0.5$ the environment is deemed *Deceptive* and is called a *stochastic compulsive liar*.

This paper describes a novel learning strategy by which the unknown parameter can be learned in both environments. To the best of our knowledge, our results are the first reported results which are applicable to the latter scenario. Another main contribution of this paper is that the proposed scheme is shown to operate equally well even when the learning mechanism is unaware whether the environment is Informative or Deceptive. The learning strategy proposed herein, called CPL–ATS, partitions the search interval into three sub-intervals, evaluates the location of the unknown point with respect to these sub-intervals using fast-converging ϵ-optimal L_{RI} learning automata and prunes the search space in each iteration by eliminating at least one partition. The CPL-ATS algorithm is shown to be provably converging to the unknown point to an arbitrary degree of accuracy with probability as close to unity as desired. Comprehensive experimental results confirm the fast and accurate convergence of the search for a wide range of values for the environment's feedback accuracy parameter p. The above algorithm can be used to learn parameters for non-linear optimization techniques.

Keywords: Learning Automata, Statistical Learning, Parameter Estimation, Stochastic Optimization, Pattern Recognition.

1. Introduction

Consider the problem of a robot (algorithm, learning mechanism) moving along the real line attempting to locate a particular point $\lambda^\star$. To assist the mechanism, we assume that it can communicate with an Environment ("Oracle") which guides it with information regarding the direction in which it should go. If the Environment is deterministic the problem is the "Deterministic Point Location Problem" which has been studied rather thoroughly. In its pioneering version Baeza-Yates *et. al* (1988) has presented the problem in a setting such that the Environment could charge the robot a cost which was proportional to the distance it was from the point sought for. The question of having multiple communicating robots locate a point on the line has also been studied by Baeza-Yates *et. al* (1988, 1992). In the stochastic version of this problem (Narendra & Thathachar, 1989), the learning mechanism attempts to locate a point in an interval with stochastic (i.e., possibly erroneous) instead of deterministic responses from the environment. Thus when it should really be moving to the "right" it may be advised to move to the "left" and vice versa with a non-zero probability.

In many optimization solutions – for example in image processing, pattern recognition and neural computing (Kashyap & Oommen, 1983; Oommen, ; Pao, 1989; Pavlidis, 1977; Press et al., 1986; Rao, 1984; Wasserman, 1989; Williams,), the algorithm works its way iteratively from its current solution to the optimal solution. Such algorithms typically have a key parameter that determines the convergence of the algorithm to the optimum. The choice of the value for this parameter is therefore critical to the algorithm. In many cases the parameter of the scheme is related to the second derivative of the criterion function, which results in a technique analogous to a Newton's root solving scheme. The disadvantages of the latter are well known – if the starting point of the algorithm is not well chosen, the scheme can diverge; in addition, 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 (Press et al., 1986; Rao, 1984; Wasserman, 1989). We suggest that our strategy to solve the stochastic point location problem can be invoked to learn the best parameter to be used in any given optimization algorithm.

2. The Stochastic Point Location Problem

The goal of the learning mechanism is to determine the optimal value of some parameter $\lambda^\star \in [0, 1)$. Although the mechanism does not know the value of $\lambda^\star$, we assume that it has responses from an intelligent environment E which is capable of informing it whether the current estimate $\hat{\lambda}$ 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, E may tell us to increase $\hat{\lambda}$ when it should be decreased, and vice versa with a *fixed* non-zero probability $1 - p$. Note that the quantity p reflects on the "effectiveness" of the environment, E . Thus, whenever $\hat{\lambda} < \lambda^\star$, the environment correctly suggests that we increase $\hat{\lambda}$ with probability p. It could as well have incorrectly recommended that we decrease $\hat{\lambda}$ with probability $1 - p$. Similarly, whenever $\hat{\lambda} > \lambda^\star$, the environment tells us to decrease $\hat{\lambda}$ with probability p, and to increase it with probability $(1 - p)$.

We further distinguish between two types of environments – *Informative* and *Deceptive*. An environment is said to be “Informative” if the probability p of it giving a correct feedback is greater than 0.5. If $p < 0.5$, the environment is said to be “Deceptive”. Thus a Deceptive environment is more likely to give erroneous feedback than a correct feedback. When the probability of correct feedback p of the environment tends to zero, the environment is deemed to be a *compulsive liar*.

3. Related Work

Oommen (Oommen) proposed and analyzed an algorithm that operates by discretizing the search space while interacting with an *Informative* environment. This algorithm takes advantage of the limited precision available in practical implementations to restrict the probability of choosing an action to only finitely many values from the interval $[0, 1)$. Its main drawback is that the steps are always very conservative. If the step size is increased the scheme converges faster, but the accuracy is correspondingly decreased. Bentley and Yao (1976) solved the deterministic point location problem of searching in an unbounded space by examining points $f(i)$ and $f(i+1)$ at two successive iterations between which the unknown point lies and doing a binary search between these points. Although it may appear that similar binary search can be applied in the stochastic point location problem, the faulty nature of the feedback from the environment may affect the certainty of convergence of the search and hence a more sophisticated search strategy is called for. Thus, whereas in Bentley and Yao’s algorithm we could confidently discard regions of the search space, we have to now resort to stochastic methods and work so that we minimize the probability of rejecting an interval of interest. This is even more significant and sensitive when the learning mechanism is unaware whether the environment is Informative or Deceptive. A novel strategy combining learning automata and pruning was used in (Oommen and Raghunath), to search for the parameter in the continuous space when interacting with an Informative environment. In this paper we extend the results of (Oommen and Raghunath) further to operate also in the continuous space, but to work equally well for both *Informative* and *Deceptive* environments. To the best of our knowledge this is the first reported result for learning in a Deceptive environment.

Santharam *et. al* (1994) present an alternative solution to find the optimal action from an infinite number of actions in a continuous space. This can be used to locate a point using a continuous space as opposed to discretizing it. Another completely different approach to locating a point will be to partition the search space into intervals and choose the mid-point of each interval repeatedly as estimate for the unknown point and use a majority voting scheme on the feedbacks obtained to eliminate one interval. Applying Chernoff bounds would then allow to precisely compute the number of steps sufficient for ensuring correct pruning with a certain level of confidence (Pelc, 1989). The difference between such an “estimation” approach and a learning automaton approach such as ours is that in the former case, a fixed number of sampling has to be done *irrespective* of how strong the feedback is, before determining the pruning. In contrast, in a learning automata approach, when one of the actions is superior, the reinforcement mechanism ensures that this action is sampled more frequently than the other. Thus the better an action is with respect to another, the faster it’s optimality is determined. Another drawback with the simple majority voting scheme is that in each epoch it can prune at most one partition. On the other hand, our scheme allows more than two thirds of the search space to be pruned in a single epoch. A third and most compelling drawback of the estimation approach in our problem setting is that it will not be feasible to detect a point in a deceptive environment where the probability of correct feedback $p < 0.5$.

4. Continuous Point Location with Adaptive Tertiary Search

The solution presented in this paper is based on the Continuous Point Location with Adaptive Tertiary Search (CPL–ATS) strategy introduced in (Oommen & Raghunath,). The basic idea behind both the solutions is to systematically explore a current interval for the parameter. This exploration is a series of estimates, each one more accurate than the previous one. Initially, we take the mid-point of the given interval to be our estimate. We then partition the given interval into disjoint sub-intervals, eliminating at least one of the subintervals from further search and recursively searching the remaining interval until the search interval is at least as small as the required resolution of estimation. Crucial to the CPL–ATS scheme is the construction of partitions and the elimination process.

The given search interval is divided into three partitions. Each of these three partitions is independently explored using an ϵ-optimal fast converging two-action learning automaton where the two actions are those of selecting a point from the left and right half of the partition under consideration. The elimination process then utilizes the the ϵ-optimality property of the underlying automata and the monotonicity of the intervals to systematically eliminate at least one of the partitions. The resultant is a new pruned interval consisting of at most two contiguous partitions from the original three. At each stage the mid-point of the remaining interval is taken to be the estimate of $\lambda^\star$. We shall assume that the individual learning automaton used is the well-known L_{RI} scheme with parameter θ, although any other ϵ-optimal scheme can be used just as effectively.

4.1 Notations and Definitions

Let $\Delta = [\sigma, \gamma)$ s.t. $\sigma \leq \lambda^\star < \gamma$ be the current search interval containing $\lambda^\star$ whose left and right (smaller and greater) boundaries on the real line are σ and γ respectively. We partition Δ into three equi-sized disjoint partitions Δ^j, $j \in \{1,2,3\}$, such that, $\Delta^j = [\sigma^j, \gamma^j)$. To formally describe the relative locations of intervals we define an interval relational operator $\prec$ such that, $\Delta^j \prec \Delta^k$ iff $\gamma^j < \sigma^k$. Since points on the real interval are monotonically increasing, we have, $\Delta^1 \prec \Delta^2 \prec \Delta^3$. For every partition Δ^j , we define L^j and R^j as its *left* half and *right* half respectively as:

$$L^j = \{x \,|\, \sigma^j \leq x < mid(\Delta^j)\}, \text{and} R^j = \{x \,|\, mid(\Delta^j) \leq x < \gamma^j\},$$

where $mid(\Delta^j)$ is the mid point of the partition Δ^j. A point $x \in L^j$ will be denoted by x^j_L, and a point $x \in R^j$ by x^j_R.

To relate the various intervals to $\lambda^\star$ we introduce the following relational operators.

$\lambda^\star \olessthan \Delta^j$	iff	$\lambda^\star < \sigma^j$.	i.e., $\lambda^\star$ is to the left of the interval Δ^j.
$\lambda^\star \ogreaterthan \Delta^j$	iff	$\lambda^\star > \gamma^j$.	i.e., $\lambda^\star$ is to the right of the interval Δ^j.
$\lambda^\star \circledequal \Delta^j$	iff	$\sigma^j \leq \lambda^\star < \gamma^j$.	i.e., $\lambda^\star$ is contained in the interval Δ^j.
$\lambda^\star \circledleq \Delta^j$	iff	$\lambda^\star \olessthan \Delta^j$ or $\lambda^\star \circledequal \Delta^j$	i.e., $\lambda^\star$ is either to the left of or inside the interval Δ^j.
$\lambda^\star \circledgeq \Delta^j$	iff	$\lambda^\star \ogreaterthan \Delta^j$ or $\lambda^\star \circledequal \Delta^j$	i.e., $\lambda^\star$ is either to the right of or inside the interval Δ^j.

These operators can be shown to satisfy the usual laws of transitivity.

4.2 Construction of the Learning Automata

In the CPL–ATS strategy, with each partition Δ^j we associate a 2-action L_{RI} automaton $\mathcal{A}^j$ $(\Sigma^j, \Pi^j, \Gamma^j, \Upsilon^j, \Omega^j)$ where, Σ^j is the set of actions, Π^j is the set of action probabilities, Γ^j is the set of feedback inputs from the environment, Υ^j is the set of action probability updating rules and Ω^j is the set of possible decision outputs of the automaton at the end of each epoch. The environment E is characterized by the probability of correct response p which is later mapped to the penalty probabilities c_k^j for the two actions of the automaton $\mathcal{A}^j$. The overall search strategy CPL–ATS, in addition uses a decision table Λ to prune the search interval by comparing the output decisions Ω^j for the three partitions. Thus $\mathcal{A}^j$, $j \in \{1, 3\}$, together with E and Λ completely define the CPL–ATS strategy.

1. *The set of actions of the automaton*(Σ^j)
   The two actions of the automaton are $\alpha_{k=0,1}^j$, where, α_0^j corresponds to selecting the *left* half L^j of the partition Δ^j, and α_1^j corresponds to selecting the *right* half R^j.

2. *The action probabilities*(Π^j)
   $P_k^j(n)$ represent the probabilities of selecting action $\alpha_{k=0,1}^j$ at step n. Initially, $P_k^j(0) = 0.5$, for $k = 0, 1$.

3. *The feedback inputs from the environment to each automaton* (Γ^j)
   It is important to recognize a subtle, but crucial point in the construction of the learning automata in CPL–ATS. From the automaton's point of view, the two actions are those of selecting either the left or the right half from its partition. However, from the environment's point of view, the automaton presents a current estimate $\hat{\lambda}$ for the true value of $\lambda^\star$, and it gives a feedback based on the relative position (or direction) of $\hat{\lambda}$ with respect to $\lambda^\star$. Thus there is a need to map the intervals to a point value and the feedback on the point value to the feedback on the choice of the intervals.

   Let the automaton select either the left or right half of the partition and then pick a point randomly (using continuous uniform probability distribution) from this sub-interval which is presented as the current estimate for $\lambda^\star$. Then the feedbacks $\beta(n)$ at step n are defined by the conditional probabilities,

$$\begin{aligned} Pr[\beta(n) = 0 \,|\, x_L^j \in L^j \text{ and } x_L^j \geq \lambda^\star] &= p \\ Pr[\beta(n) = 1 \,|\, x_L^j \in L^j \text{ and } x_L^j < \lambda^\star] &= p \\ Pr[\beta(n) = 0 \,|\, x_R^j \in R^j \text{ and } x_R^j < \lambda^\star] &= p \\ Pr[\beta(n) = 1 \,|\, x_R^j \in R^j \text{ and } x_R^j \geq \lambda^\star] &= p \end{aligned} \tag{1}$$

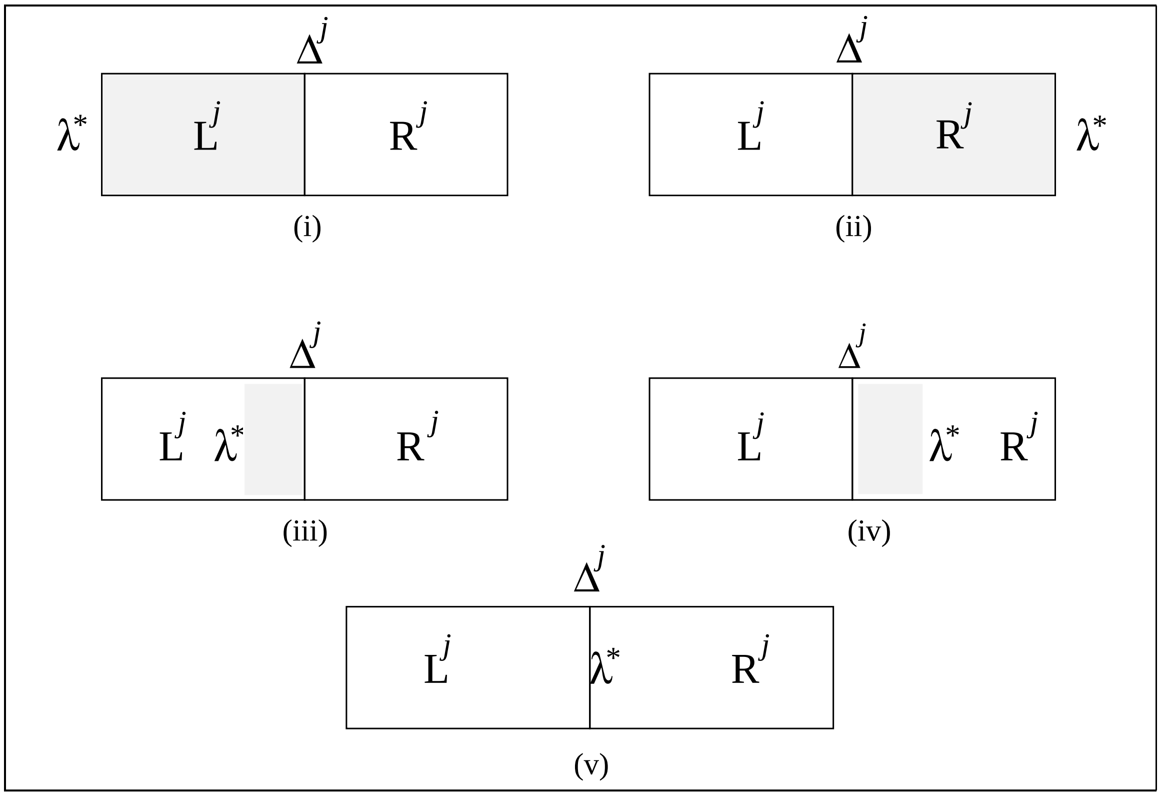

Figure 1: Relative location of $\lambda^\star$ and the convergence of the automaton.

Note that, the condition $x_L^j \in L^j$ indicates that the action α_0^j was selected and the condition $x_R^j \in R^j$ indicates the other action α_1^j was selected.

Figure 1 illustrates the feedback from the environment for different locations of $\lambda^\star$ with respect to the partition. In this figure, the shaded region represents those points which when chosen as current estimates will be rewarded by the environment with probability p. In part (i) of figure 1, $\lambda^\star$ is completely to the left of the partition Δ^j and consequently, all the points in the left half L_j will be rewarded withe probability p. Similarly, in part (ii) where $\lambda^\star$ is completely to the right of the partition Δ^j, all the points in the right half will be rewarded with probability p. Note that in these two cases, the entire half is shaded indicating that the effective probability of rewarding the selection of the half closer to $\lambda^\star$ is p.

In part (iii) where $\lambda^\star$ is inside the left half of the partition, only those points between $\lambda^\star$ and the mid-point of Δ^j will be rewarded with probability p. Similarly, in part (iv) those points between mid-point of Δ^j and $\lambda^\star$ will be rewarded. Thus, the effective probability of rewarding the choosing of the left half in part (iii) or the right half in part (iv) will be arrived at by scaling p by the ratio of the length of the shaded region over the length of the half interval.

However, an interesting situation arises when $\lambda^\star$ is exactly at the mid-point of the partition Δ^j. In this case, any point chosen from either the left or right half of the partition Δ^j will be penalized with probability $1-p$. Thus, when $p > 0.5$, even after a large number of iterations, the action probability of neither the left nor right action would have been found to have converged to unity. However, this situation occurs with probability zero.

4. *The action probability updating rules* (Υ^j)
First of all, since we are using the L_{RI} scheme, we ignore all the penalty responses. Upon reward, we obey the following updating rule :
If $\alpha^j_{k=0,1}$ was rewarded then,
$$P^j_{1-k}(n+1) \leftarrow \theta . P^j_{1-k}(n)$$
$$P^j_k(n+1) \leftarrow 1 - \theta . P^j_{1-k}(n)$$
where $0 \ll \theta < 1$ is the L_{RI} reward parameter.

5. *The decision outputs at each epoch* (Ω^j)
From the action probabilities we infer the decision Ω^j of the L_{RI} automaton $\mathcal{A}^j$, after a fixed number N_∞, of iterations. Typically, N_∞ is chosen so as to be reasonably sure that the automaton has converged. Ω^j indicates that the automaton has inferred whether $\lambda^\star$ is to the *left*, *right* or *inside* the partition. The set of values that Ω^j can take and the preconditions are given by:
$$\Omega^j = \begin{cases} \mathit{Left} & \text{if} \quad P^j_0(N_\infty) \geq 1-\epsilon. \\ \mathit{Right} & \text{if} \quad P^j_1(N_\infty) \geq 1-\epsilon. \\ \mathit{Inside} & \text{Otherwise.} \end{cases}$$

6. *The decision table for pruning the search space* (Λ)
Once the individual automata for the three partitions have made a decision regarding where they reckon $\lambda^\star$ to be, the CPL–ATS reduces the size of the search interval by eliminating at least one of the three partitions. The new pruned search interval Δ^{new} for the subsequent learning phase (epoch) is generated according to the pruning decision table Λ shown in Table 1.

Theorem 1. *If the automaton $\mathcal{A}^j$=($\Sigma^j, \Pi^j, \Gamma^j, \Upsilon^j, \Omega^j$) interacts with the environment E and gets feedbacks obeying equation(1), then the effective penalty probabilities $c^j_{k=0,1}$ for the two actions $\alpha^j_{k=0,1}$ are given by:*
$$c^j_0 = (1-p) + (2p-1).Pr(x^j_L < \lambda^\star \mid \alpha^j_0 \textit{ was chosen}) \quad (2)$$
$$c^j_1 = p - (2p-1).Pr(x^j_R < \lambda^\star \mid \alpha^j_1 \textit{ was chosen}) \quad (3)$$

Proof. By definition of the penalty probability we have,
$$\begin{aligned} c^j_0 &= Pr(\beta(n) = 1 \mid \text{sub-interval } L^j \text{ is chosen at step } n) \\ &= Pr(\beta(n) = 1 \mid x^j_L \in L^j, x^j_L < \lambda^\star).Pr(x^j_L < \lambda^\star \mid x^j_L \in L^j) + \\ &\quad Pr(\beta(n) = 1 \mid x^j_L \in L^j, x^j_L \geq \lambda^\star).Pr(x^j_L \geq \lambda^\star \mid x^j_L \in L^j) \\ &= p.Pr(x^j_L < \lambda^\star \mid x^j_L \in L^j) + (1-p).(1 - Pr(x^j_L < \lambda^\star \mid x^j_L \in L^j)) \text{ (by eq.(1)).} \end{aligned}$$

Ω^1	Ω^2	Ω^3	**New Sub-interval** Δ^{new}
$Left$	$Left$	$Left$	Δ^1
$Inside$	$Left$	$Left$	Δ^1
$Right$	$Left$	$Left$	$\Delta^1 \cup \Delta^2$
$Right$	$Inside$	$Left$	Δ^2
$Right$	$Right$	$Left$	$\Delta^2 \cup \Delta^3$
$Right$	$Right$	$Inside$	Δ^3
$Right$	$Right$	$Right$	Δ^3

Table 1: Decision table(Λ) to prune the search space based on the automata outputs Ω^j.

After some simplification, we get,

$$\begin{aligned} c_0^j &= (1-p) + (2p-1).Pr(x_L^j < \lambda^\star \,|\, x_L^j \in L^j) \\ &= (1-p) + (2p-1).Pr(x_L^j < \lambda^\star \,|\, \alpha_0^j \text{ was chosen}). \end{aligned}$$

Similarly,

$$\begin{aligned} c_1^j &= Pr(\beta(n) = 1 \,|\, \text{subinterval } R^j \text{ is chosen at step } n) \\ &= Pr(\beta(n) = 1 \,|\, x_R^j \in R^j, x_R^j \geq \lambda^\star).Pr(x_R^j \geq \lambda^\star \,|\, x_R^j \in R^j) + \\ &\quad Pr(\beta(n) = 1 \,|\, x_R^j \in R^j, x_R^j < \lambda^\star).Pr(x_R^j < \lambda^\star \,|\, x_R^j \in R^j) \\ &= p.(1 - Pr(x_R^j < \lambda^\star \,|\, x_R^j \in R^j)) + (1-p).Pr(x_R^j < \lambda^\star \,|\, x_R^j \in R^j) \text{ (by eq. (1))} \\ &= p - (2p-1).Pr(x_R^j < \lambda^\star \,|\, x_R^j \in R^j) \\ &= p - (2p-1).Pr(x_R^j < \lambda^\star \,|\, \alpha_1^j \text{ was chosen}). \end{aligned}$$

□

By the construction of the automaton, once the left or right sub-interval is chosen, a point is picked from this interval using a *uniform*[1] probability distribution. Therefore, the cumulative probability distributions for choosing our current estimate $\hat{\lambda}$ in a selected interval are given by:

$$Pr(x_L^j < x \,|\, x_L^j \in L^j) = \begin{cases} 0 & \text{if } x < \sigma^j \\ \frac{x - \sigma^j}{mid(\Delta^j) - \sigma^j} & \text{if } \sigma^j \leq x < mid(\Delta^j) \\ 1 & \text{if } x \geq mid(\Delta^j). \end{cases} \quad (4)$$

$$Pr(x_R^j < x \,|\, x_R^j \in R^j) = \begin{cases} 0 & \text{if } x < mid(\Delta^j) \\ \frac{x - mid(\Delta^j)}{\gamma^j - mid(\Delta^j)} & \text{if } mid(\Delta^j) \leq x < \gamma^j \\ 1 & \text{if } x \geq \gamma^j \end{cases} \quad (5)$$

1. The choice of uniform distribution does not put any restriction on the nature of the environment. Instead, as designers of the automaton, we have complete freedom in the choice of this distribution. However, for simplicity, we have chosen the uniform distribution.

By substituting $\lambda^\star$ for x, in the above equations, we get the quantities, $Pr(x_L^j < \lambda^\star \,|\, x_L^j \in L^j)$ and $Pr(x_R^j < \lambda^\star \,|\, x_R^j \in R^j)$ in equations (2) and (3) respectively.

5. Convergence in an Informative Environment

Lemmas 2 and 3 essentially use the ϵ-optimality property of L_{RI} automata to prove that they produce the correct decision output for each partition under an Informative environment. Lemma 4 uses the monotonicity of the real interval to establish some restrictions on the combination of decision outputs for adjacent partitions. These are then used in Theorem 2 to prove that the decision table in Table 1 is complete for the Informative environment. Theorem 3 establishes that after elimination of one or more partitions, the remaining interval still contains $\lambda^\star$, thereby assuring convergence. It should be borne in mind that these are still probabilistic results, although the probability is shown to be potentially as close to unity as we want, provided, we choose the parameters for the L_{RI} automata appropriately.

We first state a fundamental result for L_{RI} learning schemes which we will repeatedly allude to in the rest of the paper.

Lemma 1. *An L_{RI} learning scheme with parameter $0 \ll \theta < 1$ is ϵ-optimal, whenever an optimal action exists. In other words,* $\lim_{\theta\to 1}\lim_{N\to\infty} P_k^j(N) \to 1$.

The above well known result is proved in (Lakshmivarahan, 1981; Narendra & Thathachar, 1989). Thus, we are guaranteed that the for any L_{RI} scheme with the two actions $\{\alpha_0, \alpha_1\}$, if $\exists k \in \{0, 1\}$ such that $c_k^j < c_{1-k}^j$, then the action α_k^j is optimal and for this action $P_k^j(N) \to 1$ as $N \to \infty$ and $\theta \to 1$.

Lemma 2. *For an Informative environment E, given the L_{RI} scheme with a parameter θ which is arbitrarily close to unity, the following is true:*

$$\begin{array}{ll}
\textit{If } (\lambda^\star \olessthan \Delta^j), & \textit{then } Pr(\Omega^j = \textit{Left}) \to 1. \\
\textit{If } (\lambda^\star \ogreaterthan \Delta^j), & \textit{then } Pr(\Omega^j = \textit{Right}) \to 1. \\
\textit{If } (\lambda^\star \ominus \Delta^j), & \textit{then } Pr(\Omega^j = \{\textit{Left, Inside or Right}\}) \to 1.
\end{array}$$

Proof. Consider first the case $\lambda^\star \olessthan \Delta^j$. From equation (4) and (5), we get $Pr(x_L^j < \lambda^\star) = Pr(x_R^j < \lambda^\star) = 0$. Substituting these in equations (2) and (3) we get the values $c_0^j = 1 - p$ and $c_1^j = p$. Since for an informative environment $p > 0.5$, we immediately have $c_0^j < c_1^j$. Lemma 1 then assures that for any ϵ-optimal scheme such as the L_{RI} scheme used here for each $\mathcal{A}^j$, α_0^j is the optimal action and hence $P_0^j[N_\infty] \to 1$. Similar arguments for $\lambda^\star \ogreaterthan \Delta^j$ lead to the conclusion $P_1^j(N_\infty) \to 1$.

Now consider the third case when $\lambda^\star \ominus \Delta^j$. By the definition of $\ominus$ we have, $\sigma^j \le \lambda^\star < \gamma^j$. Consider first the possibility, $\sigma^j \le \lambda^\star < mid(\Delta^j)$. In this case, by eq. (4), (5) we get

$$\begin{aligned}
c_0^j &= 1 - p + (2p - 1).\frac{\lambda^\star - \sigma^j}{mid(\Delta^j) - \sigma^j} \\
c_1^j &= p
\end{aligned}$$

Since $0.5 < p \le 1$, $(2p - 1)$ is positive. The fraction in the second term of the expression for c_0^j is strictly less than unity and hence, $c_o^j < p = c_1^j$. Thus, α_0^j is the optimal action and from Lemma 1 we conclude $P_0^j[N_\infty] \to 1$.

Similarly when $mid(\Delta^j) < \lambda^\star < \sigma^j$, we get,

$$\begin{aligned} c_0^j &= p \\ c_1^j &= p - (2p-1).\tfrac{\lambda^\star - mid(\Delta^j)}{\gamma^j - mid(\Delta^j)} \end{aligned}$$

Since $p > 0.5$, it follows that $2p - 1 > 0$. The fractional factor in the second term of the expression for c_1^j is strictly less than unity, $c_1^j < 1 - p < p = c_0^j$. Thus, α_1^j is the optimal action and from Lemma 1, we get $P_1^j[N_\infty] \to 1$.

When $\lambda^\star = mid(\Delta^j)$, from equation (4) and (5), we get $Pr(x_L^j < \lambda^\star) = 1$ and $Pr(x_R^j < \lambda^\star) = 0$ and therefore, from equations (2) and (3), we get $c_0^j = c_1^j = p$. Thus, in this case there is no optimal action for any value of p. Hence, $\epsilon < P_0^j[N_\infty], P_1^j[N_\infty] < 1 - \epsilon$.

The lemma immediately follows from the decision output rule (Ω^j) in the construction of the automaton $\mathcal{A}^j$ based on the above values for $P_k^j[N_\infty]$. □

Lemma 3. *For an Informative environment E, given the L_{RI} scheme with a parameter θ which is arbitrarily close to unity, the following is true:*

$$\begin{array}{ll} \text{If } (\Omega^j = Left) & \text{then } Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j) \to 1. \\ \text{If } (\Omega^j = Right) & \text{then } Pr(\lambda^\star \text{\textcircled{$\succcurlyeq$}} \Delta^j) \to 1. \\ \text{If } (\Omega^j = Inside) & \text{then } Pr(\lambda^\star \text{\textcircled{$=$}} \Delta^j) \to 1. \end{array}$$

Proof. Consider now the first hypothesis $\Omega^j = Left$. Since this is a given event, if we were to assign probabilities for the events $\Omega^j = Left$, $\Omega^j = Right$ and $\Omega^j = Inside$, we would have, $Pr(\Omega^j = Left) = 1$, $Pr(\Omega^j = Right) = 0$, and $Pr(\Omega^j = Inside) = 0$.

Therefore,

$$Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j) = Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j \,|\, \Omega^j = Left). \tag{6}$$

First recall that the relational operator $\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j$ is actually equivalent to either of the two possibilities $\lambda^\star \text{\textcircled{$\prec$}} \Delta^j$ and $\lambda^\star \text{\textcircled{$=$}} \Delta^j$ with the third possibility being, $\lambda^\star \text{\textcircled{$\succ$}} \Delta^j$. Now, by Baye's rule for conditional probabilities,

$$\begin{aligned} &Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j \,|\, \Omega^j = Left) = \\ &\frac{Pr(\Omega^j = Left \,|\, \lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j).Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j)}{Pr(\Omega^j = Left \,|\, \lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j).Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j) + Pr(\Omega^j = Left \,|\, \lambda^\star \text{\textcircled{$\succ$}} \Delta^j).Pr(\lambda^\star \text{\textcircled{$\succ$}} \Delta^j)} \end{aligned} \tag{7}$$

From Lemma 2, we know that $Pr(\Omega^j = Left \,|\, \lambda^\star \text{\textcircled{$\succ$}} \Delta^j) \to 0$, causing the product in the second term of the denominator to approach zero. Thus,

$$Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j \,|\, \Omega^j = Left) \to 1$$

which, when substituted in eq. (6), leads to the conclusion that $Pr(\lambda^\star \text{\textcircled{$\preccurlyeq$}} \Delta^j) \to 1$.

In exactly similar lines, appealing to Baye's rule and the results of Lemma 2, we can prove the lemma for the other two hypotheses. □

Lemma 4. *In an Informative environment E, if the CPL-ATS learning mechanism uses the same L_{RI} scheme at all levels of the recursion and a parameter θ arbitrarily close to unity, the following is true:*

$$\begin{array}{ll} \textit{If } (\Omega^j = Left) & \textit{then } Pr(\Omega^{j+1} = Left) \to 1 \\ \textit{If } (\Omega^j = Inside) & \textit{then } Pr(\Omega^{j+1} = Left) \to 1 \\ \textit{If } (\Omega^j = Right) & \textit{then } Pr(\Omega^{j+1} = \{Left,\ Right \textit{ or } Inside\}) \to 1. \end{array}$$

Proof. From Lemma 3, we know that, if $(\Omega^j = Left)$, then $Pr(\lambda^\star \circledleq \Delta^j) \to 1$. As a consequence of the the monotonicity of the search interval Δ, we have $\Delta^j \prec \Delta^{j+1}$. Thus, $Pr(\lambda^\star \circledleq \Delta^j) \to 1$ and $\Delta^j \prec \Delta^{j+1}$ together imply that $Pr(\lambda^\star ⧀ \Delta^{j+1}) \to 1$. Now, using Lemma 2 we conclude $Pr(\Omega^{j+1} = Left) \to 1$.

Invoking Lemma 3 in the same manner, we argue that if $\Omega^j = Inside$, then $Pr(\lambda^\star ⊜ \Delta^j) \to 1$ and by the monotonicity of the intervals, $\Delta^j \prec \Delta^{j+1}$, whence $Pr(\lambda^\star ⧀ \Delta^{j+1}) \to 1$. Again Lemma 2 leads us to conclude that in this case, $Pr(\Omega^{j+1} = Left) \to 1$.

Finally, if we apply Lemma 3 to the assertion $\Omega^j = Right$ we see that this is equivalent to the statement that $Pr(\lambda^\star \circledgeq \Delta^j) \to 1$. This however allows all the three possibilities, $\lambda^\star ⧀ \Delta^{j+1}$, $\lambda ⊜ \Delta^{j+1}$ and $\lambda^\star ⧁ \Delta^{j+1}$. Applying Lemma 2 to each of these, we get the result $Pr(\Omega^j = Left) \to 1$, $Pr(\Omega^j = \{Left, Inside, Right\}) \to 1$ and $Pr(\Omega^j = Right) \to 1$. Collectively, this is the same as saying, $Pr(\Omega^j = \{Left, Inside, Right\}) \to 1$. □

Theorem 2. *If the environment is Informative and if the partitions use the same L_{RI} scheme with parameters θ as close to unity as needed, then the decision table given in Table 1 is complete.*

Proof. In any decision table with three input variables and three possible values for each of these variables, we expect a total of 27 potential entries. However, Lemma 4 imposes constraints on the output value combinations of the automata. A straightforward enumeration of these possibilities will show that the only possible combinations of outputs for the three L_{RI} automata are the seven entries shown in the decision table. Consequently Table 1 is complete. □

A consequence of this theorem is that any entry not shown in the decision table is said to be *inconsistent* in the sense that for an Informative environment and appropriate θ, the probability of occurrence of this entry is arbitrarily close to zero.

Theorem 3. *If the algorithm uses the same L_{RI} scheme at all levels of the recursion with a parameter θ arbitrarily close to unity and N_∞ sufficiently large, then for an Informative environment, the unknown $\lambda^\star$ is always contained in the new search-interval Δ^{new} resulting from the application of the decision rules of Table 1.*

Proof. Consider the first row of Table 1 where we see that $\Omega^1 = Left$, $\Omega^2 = Left$ and $\Omega^3 = Left$. Appealing to Lemma 3 for each of the automata outputs, we get $Pr(\lambda^\star \circledleq \Delta^1) \to 1$, $Pr(\lambda^\star \circledleq \Delta^2) \to 1$ and $Pr(\lambda^\star \circledleq \Delta^3) \to 1$. When we consider the fact that $\Delta^1 \prec \Delta^2 \prec \Delta^3$, the above three reduce to the equivalent predicate, $Pr(\lambda^\star \circledleq \Delta^1) \to 1$. By the definition of $\circledleq$ we have the two possibilities: $\lambda^\star ⧀ \Delta^1$ and $\lambda^\star ⊜ \Delta^1$. But, since $\lambda^\star ⊜ \Delta$ and $\Delta = \Delta^1 \cup \Delta^2 \cup \Delta^3$, we can rule out $\lambda^\star ⧀ \Delta^1$. Therefore, $Pr(\lambda^\star ⧀ \Delta^1) \to 1$. Thus, the partition

Δ^1 that remains after pruning still contains $\lambda^\star$ with as high a probability as we want. The same arguments can be repeated for each of the other entries and are omitted in the interest of brevity. □

With the above results, we are ready to construct a mechanism that can learn the parameter $\lambda^\star$ in an informative environment. The formal algorithm for doing so will be presented after we extend the results to deceptive environments since we intend to use a common framework for learning $\lambda^\star$ in both these environments.

6. Convergence in a Deceptive Environment

Let E be an environment with a probability of correct feedback p. Then, another environment E* , is said to be the *dual* of the environment E , if and only if, its probability of correct feed back is $1-p$.

Lemma 5. *If $c^j_{k\in\{0,1\}}$ are the penalty probabilities for the two actions $\alpha^j_{k\in\{0,1\}}$ of an automaton $\mathcal{A}^j$ that operates in an environment E with probability of correct feedback p, then, the penalty probabilities $c'^j_{k\in\{0,1\}}$ of the actions $\alpha^j_{k\in\{0,1\}}$ of the same automaton $\mathcal{A}^j$ under the dual environment E* are given by the relation $c'^j_k = 1 - c^j_k$.*

Proof. Let $p' = 1-p$ be the probability of correct feedback under the dual environment E* . By equation 2, we have

$$c'^j_0 = (1-p') + (2p'-1).Pr(x^j_L < \lambda^\star \mid \alpha^j_0 \text{ was chosen})$$

Substituting $1-p$ for p' in the above equation we get,

$$\begin{aligned} c'^j_0 &= p - (2p-1).Pr(x^j_L < \lambda^\star \mid \alpha^j_0 \text{ was chosen}) \\ &= 1 - (1-p+(2p-1).Pr(x^j_L < \lambda^\star \mid \alpha^j_0 \text{ was chosen})) \\ &= 1 - c^j_0 \end{aligned}$$

In exactly the same way, we can show that $c'^j_1 = 1 - c^j_1$. □

The following lemma is a natural corollary of Lemma 5.

Lemma 6. *If α^j_k is the ϵ-optimal action for an L_{RI} automaton $\mathcal{A}^j$ under a given environment E, then α^j_{1-k} is the ϵ-optimal action under its dual environment E* and vice versa.*

Let E be the given *Deceptive* environment. By definition then, we have its probability of correct response $p < 0.5$. We now construct a dual E* of this environment with a corresponding probability of correct response $p' = 1-p$. Then this dual environment is *Informative* since, $p' > 0.5$. Thus if the learning autmaton can some how determine whether a given environment is Deceptive or Informative, then Lemmas 5 and 6 assure us that by interchanging the actions (or equivalently the penalties and rewards), the automaton will still be able to converge to the optimal action with as high a probability as we want.

Theorem 4. *Given a Deceptive environment E,*

$$\begin{array}{ll} \textit{If } (\lambda^\star \olessthan \Delta^j), & \textit{then } Pr(\Omega^j = Right) \to 1 \\ \textit{If } (\lambda^\star \ogreaterthan \Delta^j), & \textit{then } Pr(\Omega^j = Left) \to 1 \\ \textit{If } (\lambda^\star \ominus \Delta^j), & \textit{then } Pr(\Omega^j = \{Left,\ Inside \textit{ or } Right\}) \to 1. \end{array}$$

Proof. Since E is a Deceptive environment, the probability of it giving correct responses is $p < 0.5$. Now construct a dual E* for this environment such that the probability of its correct feedback $(p' = 1 - p) > 0.5$. Clearly, E* is an Informative environment and hence we can apply Lemma 2 to it. Consequently, if we are given that $\lambda^\star \olessthan \Delta^j$, by Lemma 2, we know that for E* , $Pr(\Omega^j = Left) \to 1$, indicating thereby, that α_0^j is the optimal action. Now by Lemma 5, under E we will have α_1^j to be the optimal action. Therefore, we conclude that for E, $Pr(\Omega^j = Right) \to 1$. Identical arguments apply for the second hypothesis as well. When the automaton is inconclusive, i.e., $\Omega^j = Inside$, there is no optimal action under either environment and therefore, under both environments $Pr(\Omega^j = Inside) \to 1$. □

Based on Theorem 4 above, if we invert *Left* to *Right* and vice versa in the decision table Γ of Table 1, the first row $< Left, Left, Left >$ gets transformed to the last row $< Right, Right, Right >$ and vice versa and the rest of the rows will get transformed to one of the impossible entries. Thus except for the two cases where the automata outputs are all *Left* or all *Right*, we can directly detect a deceptive environment. We now provide a mechanism for dealing with the two extreme cases as well.

Theorem 5. *Suppose it is given that $\lambda^\star \ogreaterthan \Delta^1$, $\lambda^\star \ominus \Delta^2$ and $\lambda^\star \olessthan \Delta^3$. Then under a Deceptive environment, the decision output vector $\vec{\Omega}$ for the three automata $\mathcal{A}^j, j \in \{1,2,3\}$, will be inconsistent with the decision table of Table 1. Conversely, if for the given environment and $\lambda^\star$ as above, the decision output vector $\vec{\Omega}$ of the automata is inconsistent with the decision table, then the environment is Deceptive.*

Proof. Applying Theorem 4 to each of $\lambda^\star \ogreaterthan \Delta^1$, $\lambda^\star \ominus \Delta^2$ and $\lambda^\star \olessthan \Delta^3$, we have,

$$\begin{array}{l} Pr(\Omega^1 = Left) \to 1 \\ Pr(\Omega^2 = \{Left,\ Inside \textit{ or } Right\}) \to 1 \\ Pr(\Omega^3 = Right) \to 1 \end{array}$$

By inspecting the decision table Γ, (Table 1), we see that there are no entries for this output vector $\vec{\Omega} = [Left, \{Left,\ Inside \textit{ or } Right\}, Right]$ and hence the entry is inconsistent with Table 1. The converse is proved by contradiction by alluding to the completeness result of Theorem 2 for an Informative environment. Therefore, whenever the decision output vector $\vec{\Omega}$ is inconsistent with Table 1, we can safely conclude that the environment is Deceptive. □

Theorem 6. *Let $\mathcal{I}= [0,1)$ be the original search interval in which $\lambda^\star$ is to be found. Let $\mathcal{I}' = [-1,2)$ be the initial search interval used by CPL–ATS. Then, CPL–ATS always determines whether or not an environment is Deceptive after a single epoch.*

Proof. First of all, we can see that $\lambda^\star \in \mathcal{I}'$ because, $\mathcal{I} \subset \mathcal{I}'$. When we divide $\mathcal{I}'$ into three equal partitions we get, $\Delta^1 = [-1,0)$, $\Delta^2 = [0,1)$ and $\Delta^3 = [1,2)$. Since $\lambda^\star \in \mathcal{I} = \Delta^2$, we have, $\lambda^\star \ogreaterthan \Delta^1$, $\lambda^\star \ominus \Delta^2$, $\lambda^\star \olessthan \Delta^3$, which is the pre-condition for Theorem 5. Hence, by appealing to Theorem 5 we see that if the environment was Deceptive, we would get an inconsistent decision vector. If not, by Theorem 2 we would get a consistent decision vector. Thus, after *one* epoch we conclude decisively about the nature of the environment. □

Theorem 6 gives us a simple mechanism for detecting Deceptive environments. Thus, we start with an initial search interval $\mathcal{I}' = [-1, 2)$, partition it into three sub-intervals and run the three L_{RI} automata $\mathcal{A}^j, j = 1, 2, 3$ for exactly *one* epoch. At the end of the epoch, we use Table 1 to check if the decision vector $\vec{\Omega}$ has a valid entry or not and accordingly conclude that the environment is Informative or Deceptive. If the environment was found to be Deceptive, we flip the probability update rules; i.e., essentially treat every reward as a penalty and vice versa. Lemma 5 guarantees that we will then converge to the optimal action. If instead, the environment was found to be Informative we proceed with the search.

Note that the expanded interval $\mathcal{I}'$ is needed only for the first epoch, to detect the environment's nature. Once this is detected, we use the original interval $\mathcal{I}$ to search for $\lambda^\star$. It is assumed that the fact that we use the expanded intervals $[-1, 0)$ and $[1, 2)$ does not affect the responses given by the environment.

7. Implementation and Evaluation of CPL–ATS Scheme

The CPL–ATS strategy is fairly simple to implement because, it uses a straight forward partitioning of the search interval and a simple decision table for elimination, besides using the well known L_{RI} learning algorithm. In this section we present the pseudo code for the overall learning strategy as well as that of the L_{RI} learning algorithm. We also present a sample trace to demonstrate the correct convergence under a *Deceptive* environment. Finally, we present numerical results to augment our analysis presented in the previous sections, both for informative and deceptive environments.

7.1 Implementation of the CPL–ATS Strategy

The CPL–ATS strategy has been implemented and tested with a wide range of inputs. The pseudo-code for the algorithms and a sample trace are presented in figures 2 to 3 to illustrate the workings of the CPL–ATS strategy.

Observe that in Figure 2, the algorithm starts with an initial interval $[-1, 2)$ instead of the given original interval. Also note that the algorithm executes for one epoch for each of the partitions of this interval and determines the nature of the environment as to whether it is Informative or Deceptive. Finally, note that in Procedure ExecuteL_{RI} of Figure 7.1, rewards and penalties awarded by the environment are interchanged if the environment is *Deceptive*. The rationale for these steps were earlier explained in section 6.

Figure 3 shows the trace of execution of the CPL–ATS algorithm. In this example run, the initial search interval is $[0, 1)$ and $\lambda^\star$ is 0.9123. The environment is Deceptive with $p = 0.1$. The search terminates when the width of the interval is ≤ 0.01. The reward factor θ of the automata is 0.8 and $\epsilon = 0.005$. In every invocation of CPL–ATS the results of the automata are given as a set of decision outputs Ω^1, Ω^2 and Ω^3 and are determined after the L_{RI} scheme has run for $N_\infty = 250$ iterations. Note that at step 10 in Figure 3, the algorithm terminates when the width of the interval [0.906, 0.915] is less than the specified

```
**Algorithm CPL–ATS**($\Delta^{Original}$)

**Input :** $p$, $\theta$, $\epsilon$, *Resolution.*

**Output :** The final estimate $\hat{E}(\lambda(N_\infty))$.

**Method**
**Begin**
        $\Delta := [-1, 2)$; $\Delta^{Original} = [0, 1)$
        $\Delta^1 := [-1, 0)$; $\Delta^2 := [0, 1)$; $\Delta^3 := [1, 2)$;
        EnvironType := *Unknown*;
        **For** $j := 1$ **To** 3 **Do**
              $\Omega^j$ := Execute$L_{RI}$($j$, EnvironType)
        **EndFor**
        $\vec{\Omega} := [\Omega^j, j = 1, 2, 3]$
        $\Delta^{new}$ := ChooseNewSearchInterval($\vec{\Omega}$, Decision-Table)
        **If** ($\Delta^{new} = \Delta^{Original} = [0, 1)$) **Then**
            Search($\Delta^{Original}$, *Deceptive)*
        **Else**
            Search($\Delta^{Original}$, *Informative)*
        **EndIf**
**END Algorithm CPL–ATS**


**Procedure Search**($\Delta$, **EnvironType**)

**Input :** *Resolution:* the size of the smallest significant interval containing $\lambda^\star$. Its magnitude
        determines the accuracy of the final estimate and is used to terminate the recursion.

**Output :** The estimate of $\lambda^\star$. It is derived as the mid-point of the final search interval.

**Method**
**Begin**
        **If** (WidthOfInterval($\Delta$) $\leq$ Resolution) **Then**
            **Return** (MidPointOfInterval($\Delta$))                        /* Terminate Recursion */
        **Else**
            [$\Delta^1$, $\Delta^2$, $\Delta^3$] := PartitionInterval ($\Delta$)
            **For** $j := 1$ **To** 3 **Do**
                  $\Omega^j$ := Execute$L_{RI}$($j$, EnvironType)
            **EndFor**
            $\vec{\Omega} := [\Omega^j, j = 1, 2, 3]$
            $\Delta^{new}$ := ChooseNewSearchInterval($\vec{\Omega}$, Decision-Table)
            Search($\Delta^{new}$, EnvironType)                                   /* Tail Recursion */
        **EndIf**
**END Procedure Search**
```

Figure 2: Algorithm CPL–ATS: Detection of the Nature of the Environment and the Overall Search Strategy.

```
Procedure Execute$L_{RI}$(j, EnvironType)
Input: The sub-interval, $\Delta^j$; the parameters $\theta$ and $\epsilon$ of the $L_{RI}$ scheme; the parameter $p$ of the
    environment.
Output: A decision from the set {Left, Right, Inside} representing whether the automaton has
    determined $\lambda^\star$ to be to the left, right or inside the current partition.

Method
Begin
    For $k := 0$ To 1 Do $P_k^j := 0.5$ EndFor
    For $i := 1$ To $N_\infty$ Do
        $k :=$ ChooseAction($\Delta^j$)
        If ($k = 0$) Then                                        /* $\alpha_0^j$ is the chosen action */
            $x_L^j :=$ PickARandomPointIn($L^j$)                 /* $L^j$ is the left half of $\Delta^j$ */
            $\beta :=$ GetFeedBack($k$, $x_L^j$)                    /* Compare $x_L^j$ with $\lambda^\star$ */
            /* Flip Reward and Penalty if environment is thought to be deceptive */
            If (EnvironType = Deceptive) Then $\beta := 1 - \beta$ EndIf
            If ($\beta = 0$) Then                             /* Oracle has rewarded the choice */
                $P_1^j := \theta.P_1^j$; $P_0^j := 1 - P_1^j$
            EndIf
        Else                                                     /* $\alpha_1^j$ is the chosen action */
            $x_R^j :=$ PickARandomPointIn($R^j$)                /* $R^j$ is the right half of $\Delta^j$ */
            $\beta :=$ GetFeedBack($k$, $x_R^j$)                    /* Compare $x_R^j$ with $\lambda^\star$ */
            /* Flip Reward and Penalty if environment is thought to be deceptive */
            If (EnvironType = Deceptive) Then $\beta := 1 - \beta$ EndIf
            If ($\beta = 0$) Then                             /* Oracle has rewarded the choice */
                $P_0^j := \theta.P_0^j$; $P_1^j := 1 - P_0^j$
            EndIf
        EndIf
    EndFor
    If ($P_0^j \geq 1 - \epsilon$) Then Return (Left) EndIf
    If ($P_1^j \geq 1 - \epsilon$) Then Return (Right) EndIf
    Return (Inside)
End Procedure Execute$L_{RI}$
Procedure ChooseNewSearchInterval ($\vec{\Omega}$), Decision-Table
Input: The Results of the $L_{RI}$ Automata $\mathcal{A}^j$
Output: The $\Delta^{new}$, the new sub-interval to be processed.

Method
        If ($\vec{\Omega} \in$ Table 1) Then
            Return (NewSubInterval(Table 1, $\vec{\Omega}$))
        Else
            Return ($\Delta^{Original}$)
        EndIf
End Procedure ChooseNewSearchInterval
```

p	$\theta = 0.8$	$\theta = 0.85$	$\theta = 0.9$
0.10	0.912298	0.912273	0.912194
0.15	0.912312	0.912298	0.912222
0.20	0.912193	0.912299	0.912236
0.80	0.912317	0.912284	0.912234
0.85	0.912299	0.912275	0.912202
0.90	0.912302	0.912275	0.912202

Table 2: Asymptotic value of $\hat{E}[\lambda(N_\infty)]$ as it varies with p and θ. In all the cases, $\lambda^\star = 0.9123$, $N_\infty = 250$ and $\epsilon = 0.005$. The values shown are averaged over 50 independent experiments.

resolution (0.01). The estimated value for $\lambda^\star$ is the mid-point of the interval $[0.906, 0.915]$ which is 0.9105.

7.2 Experimental Results

The parameter learning mechanism, CPL–ATS, described in this paper was experimentally evaluated to verify the validity of our analytic results and to examine its rate of convergence. To verify the power of the scheme and to study its effectiveness for various conditions, simulation experiments were conducted for various values of θ , the reward factor of the L_{RI} automata, and for various values of p, the probability of the environment correctly providing the feed back. In all the experiments it was assumed that $\lambda^\star \in [0, 1)$. In each case, a single screening epoch was run using the expanded interval $[-1, 2)$ to detect whether or not the environment is Informative. After that, the given original search interval $[0, 1)$ was used as the starting point. Each epoch consisted of 250 iterations (N_∞) of the three L_{RI} automata. At the end of each epoch the decision table Table 1 was consulted to prune the current search interval and the algorithm was recursively evoked. The recursion was terminated when the width of the interval was less than twice the desired accuracy.

The results of our experiments are truly conclusive and confirm the power of the CPL-ATS scheme. Although several experiments were conducted using various $\lambda^\star$ and parameter values, we report for brevity sake, only the results for $\lambda^\star = 0.9123$. For this value, several independent replications with different random number streams were performed to minimize the variance of the reported results. The reported results are averaged over the replications.

The mean asymptotic value of the estimate for $\lambda^\star$ are shown in Table 2 for various values of p and θ. The final estimates agree with the true value 0.9123 of $\lambda^\star$to the first three decimal places for *all* values of p shown in the table. Figure 4 shows the convergence of the algorithm for $p = 0.1, 0.35$ and 0.8. This figure plots the running estimate at the end of each epoch. The values shown are actually the averaged over 50 replications for each set of parameters.

We first note that the convergence of the algorithm is almost identical for $p = 0.1$ and $p = 0.8$ even though the former represents a highly unreliable environment whereas the

Step 0 :$\Delta = [-1, 2]$

Partitions: $\Delta^1 = [-1, 0]$ $\Delta^2 = [0, 1]$ $\Delta^3 = [1, 2]$

Results: $\Omega^1 = Left$ $\Omega^2 = Left$ $\Omega^3 = Right$

Conclusion:There is no Decision Table entry $< Left, Left, Right >$.
Mark Environment as *Deceptive* and invert penalties and rewards;
Reset the search interval to $[0, 1)$.

Step 1:$\Delta = [0.0, 1.0]$

Partitions: $\Delta^1 = [0.0, 0.333]$ $\Delta^2 = [0.333, 0.667]$ $\Delta^3 = [0.667, 1.0]$

Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Right$

Conclusion: New Search Interval is $\Delta^3 = [0.667, 1.0]$

Step 2:$\Delta = [0.667, 1.0]$

Partitions: $\Delta^1 = [0.667, 0.778]$ $\Delta^2 = [0.778, 0.889]$ $\Delta^3 = [0.889, 1.0]$

Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^2 \cup \Delta^3 = [0.778, 1.0]$

Step 3:$\Delta = [0.778, 1.0]$

Partitions:$\Delta^1 = [0.778, 0.852]$ $\Delta^2 = [0.852, 0.926]$ $\Delta^3 = [0.926, 1.0]$

Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^2 \cup \Delta^3 = [0.852, 1.0]$

Step 4:$\Delta = [0.852, 1.0]$

Partitions:$\Delta^1 = [0.852, 0.901]$ $\Delta^2 = [0.901, 0.951]$ $\Delta^3 = [0.951, 1.0]$

Results: $\Omega^1 = Right$ $\Omega^2 = Left$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^1 \cup \Delta^2 = [0.852, 0.951]$

Step 5:$\Delta = [0.852, 0.951]$

Partitions:$\Delta^1 = [0.852, 0.885]$ $\Delta^2 = [0.885, 0.918]$ $\Delta^3 = [0.918, 0.951]$

Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^2 \cup \Delta^3 = [0.885, 0.951]$

Step 6:$\Delta = [0.885, 0.951]$

Partitions:$\Delta^1 = [0.885, 0.907]$ $\Delta^2 = [0.907, 0.929]$ $\Delta^3 = [0.929, 0.951]$

Results: $\Omega^1 = Right$ $\Omega^2 = Left$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^1 \cup \Delta^2 = [0.885, 0.929]$

Step 7:$\Delta = [0.885, 0.929]$

Partitions:$\Delta^1 = [0.885, 0.899]$ $\Delta^2 = [0.899, 0.914]$ $\Delta^3 = [0.914, 0.929]$

Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^2 \cup \Delta^3 = [0.899, 0.929]$

Step 8:$\Delta = [0.899, 0.929]$

Partitions:$\Delta^1 = [0.899, 0.909]$ $\Delta^2 = [0.909, 0.919]$ $\Delta^3 = [0.919, 0.929]$

Results: $\Omega^1 = Right$ $\Omega^2 = Left$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^1 \cup \Delta^2 = [0.899, 0.919]$

Step 9:$\Delta = [0.899, 0.919]$

Partitions:$\Delta^1 = [0.899, 0.906]$ $\Delta^2 = [0.906, 0.912]$ $\Delta^3 = [0.912, 0.919]$

Results: $\Omega^1 = Right$ $\Omega^2 = Right$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^2 \cup \Delta^3 = [0.906, 0.919]$

Step 10:$\Delta = [0.906, 0.919]$

Partitions:$\Delta^1 = [0.906, 0.910]$ $\Delta^2 = [0.910, 0.915]$ $\Delta^3 = [0.915, 0.919]$

Results: $\Omega^1 = Right$ $\Omega^2 = Left$ $\Omega^3 = Left$

Conclusion: New Search Interval is $\Delta^1 \cup \Delta^2 = [0.906, 0.915]$

Figure 3: Trace of execution of an example run of CPL–ATS algorithm.

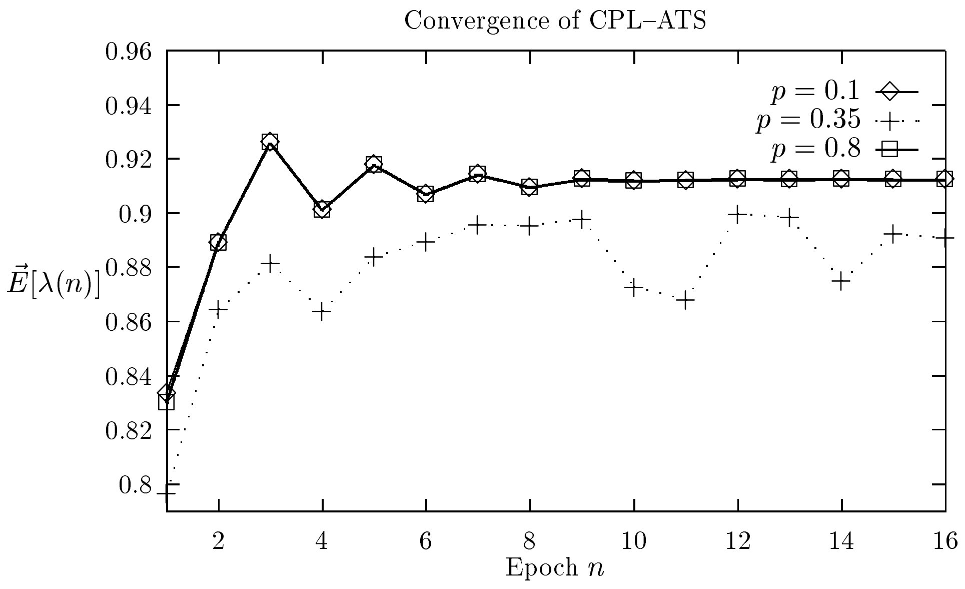

Figure 4: Convergence of estimate $\hat{E}[\lambda(n)]$ for $\theta = 0.8$. The value shown are averaged over 50 replications. The unknown parameter $\lambda^\star = 0.9123$.

latter is a highly reliable environment. This is not surprising because, after having detected in epoch 0 that the environment is not Informative, the CPL–ATS strategy switches the reward and penalty feedbacks, effectively behaving as $p' = 1 - p$. Thus, even in the very first epoch a value of 0.83 is achieved which is within 9 percent error of the true value of $\lambda^\star = 0.9123$. In two more epochs, for all the four p values shown, the estimate is 0.925926 which is within 1.5 percent error. Note however, that for $p = 0.35$ the convergence is relatively sluggish. It took 25 epochs for it to reach the terminal value of 0.906453 which is within 0.64 percent of the true $\lambda^\star$. Thus, as p is increasingly closer to 0.5, θ must be set close to unity and N_∞ (the number of iterations per epoch) has to be increased to achieve convergence.[2] Thus, a p value very close to 0.5 represents a highly inconsistent feedback wherein half the time the environment directs the learning mechanism in one direction and the rest of the time in another.

8. Conclusions

In this paper we have considered the problem of a learning mechanism locating a point on a line when it is interacting with a random stationary environment which essentially informs it, possibly erroneously, which way it should move to reach the point being sought. The first reported paper to solve this problem (Oommen,) presented a solution which operated in a discretized space. Later in (Oommen & Raghunath,), we presented a new scheme by which

2. For the parameter values that we experimented, we found that the algorithm consistently converges for p in the ranges $(0, 0.4)$ and $(0.6, 1.0)$.

the point can be learned using a combination of various learning principles. The heart of this strategy involved performing a controlled random walk on the underlying space using the L_{RI} updating rule and then intelligently pruning the space using an adaptive tertiary search. The overall learning scheme is shown to be ϵ-optimal. We have also shown that the results of (Oommen & Raghunath,) can also be generalized to the cases when the environment is a stochastic compulsive liar (i.e., is Deceptive). Although the problem is solved in its generality, its application in non-linear optimization has also been suggested. 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 (Pao, 1989; Wasserman, 1989; Oommen,).

References

Baeza-Yates, R. A., Culberson, J. C., & Rawlins, G. J. E. (1988). Searching with uncertainty. In *Proceedings of the Scandinavian Workshop on Algorithms and Theory, SWAT 88*, pp. 176–189.

Baeza-Yates, R. A., & Schott, R. (1992). Parallel searching in the plane. In *1992 International Conference of the Chilean Computer Society, IC-SCCC 92*. Chilean Computer Society.

Bentley, J. L., & Yao, A. C.-C. (1976). An almost optimal algorithm for unbounded searching. *Information Processing Letters*, 82–87.

Kashyap, R., & Oommen, B. (1983). Scale preserving smoothing of polygons. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 667–671.

Lakshmivarahan, S. (1981). *Learning Algorithms Theory and Applications.* Springer-Verlag.

Narendra, K., & Thathachar, M. (1989). *Learning Automata.* Prentice-Hall.

Oommen, B. Stochastic searching on the line and its applications to parameter learning in non-linear optimization. *To appear in IEEE Transactions on Systems, Man and Cybernetics.*

Oommen, B., & Raghunath, G. Intelligent tertiary searching for stochastic point location. *To appear in IEEE Transactions on Systems, Man and Cybernetics.*

Pao, Y.-H. (1989). *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley, Reading, Mass.

Pavlidis, T. (1977). *Structural Pattern Recognition.* Springer-Verlag, NewYork.

Pelc, A. (1989). Searching with known error probability. *Theoretical Computer Science*, *63*, 185–202.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1986). *Numerical Recipes : The Art of Scientific Computing.* Cambridge University Press, Cambridge.

Rao, S. (1984). *Optimization: Theory and Applications* (2nd. ed. edition). John Wiley, New Delhi.

Santharam, G., Sastry, P., & Thathachar, M. (1994). Continuous action set learning automata for stochastic optimization. *Journal of the Franklin Institute*, *331B5*, 607–628.

Wasserman, P. D. (1989). *Neural Computing : Theory and Practice.* van Nostrand Reinhold, New York.

Williams, R. Simple statistical gradient-following algorithms for connectioninst reinforcement learning. *Machine Learning*, *8*, 229–256.