*J. Indian Inst. Sci.*, Jan.–Feb. 1997, 77, 15–45.


# A parallel stochastic algorithm for learning logic expressions under noise

K. RAJARAMAN[†] AND P. S. SASTRY[‡]
Dept. of Electrical Engineering Indian Institute of Science Bangalore - 560 012, India.

**Abstract**

The problem of learning conjunctive and disjunctive concepts from a series of positive and negative examples of the concept is considered. Formulating the problem in the Probably Approximately Correct(PAC) Learning framework, the goal of such inductive learning is precisely characterized. A parallel distributed stochastic algorithm is presented. It has been proved that the algorithm learns the class of simple conjunctive concepts (in the presence of upto 50% unbiased noise) over both nominal and linear attributes. As an extension to this, an algorithm that learns a class of disjunctive concepts is proposed. Through empirical studies it is seen that the algorithm is quite efficient for learning conjunctive concepts and certain disjunctive concepts.

**Keywords:** Learning Automata, Games of Learning Automata, Concept Learning, PAC Learning, Pattern Recognition.

## 1. Introduction

Concept learning or learning from examples is one of the most extensively studied forms of learning in Artificial Intelligence (AI). Apart from the fact that such a study throws light on human intelligent behaviour, the problem is interesting because of its applications in areas like Expert Systems[1]. The problem of concept learning involves learning to classify the objects of a domain using a set of pre-classified objects from the domain. The pre-classified objects are supplied as positive examples, i.e. those satisfying the target concept description, and negative examples, i.e. those other than the positive examples. After seeing a set of such examples, the learning system is required to infer a concept description which can be used to classify new objects from the domain. For example, one might consider learning to classify patients as jaundice affected or not, by looking at data obtained through clinical tests. The classification problem as discussed in Pattern Recognition (PR) literature[2] is similar to concept learning. Though the common goal in both fields is to find a classification rule, concept learning differs from pattern classification in that the attributes (features) used to describe the examples need not be numerical values. For example, in our disease diagnosis problem, a possible attribute used to describe the data might be the eye-color of the patient which takes values from a non–numerical set. Also, in concept learning, we are interested in learning varied concept representations like logic expressions[3], decision trees[4], etc. rather than decision surfaces in $\mathcal{R}^n$.

[†]Current address: Information Technology Institute, 11, Science Part Road, Science Park II, Singapore 117 685, , email: kanagasa@iti.gov.sg,
[‡]email: sastry@vidyut.ee.iisc.ernet.in.

For the class of concept learning problems considered in this paper, we assume that a set of attributes characterizing the domain is specified. The examples are expressed as a vector of values assumed by these attributes. The attributes may take values in arbitrary sets on which there may or may not be any structure. By experiencing a subset of these examples, the problem is to learn the "right" concept description. This problem has been investigated in AI and Computational Learning Theory (COLT) and several algorithms have been proposed[5–9]. We discuss some of the important algorithms in Section 5.

We work within the Probably Approximately Correct (PAC) Learning framework[10–12]. We propose a parallel stochastic algorithm based on a cooperating team of learning automata. The algorithm is shown to *correctly* learn (in a sense to be defined shortly) the class of simple conjunctive concepts (defined in Section 2) expressed through nominal and/or linear attributes. Moreover, our algorithm is robust and can tackle upto 50% of unbiased classification noise in examples. It is an incremental algorithm and hence does not need to store the examples. It is also a parallel algorithm and can be implemented on a parallel distributed network of simple processors with only local computation. This paper is an extension of the work by Sastry *et.al*[13]. The algorithm in Sastry *et al*[13]. can learn concepts expressed through nominal attributes only and cannot handle linear attributes. In this paper, we present a generalized algorithm that can handle both nominal and linear attributes. Though we follow the idea of proving convergence as in the earlier version, the proof here is presented in a more simplified form. Here, we also show how we can modify the algorithm to learn one subclass of disjunctive concepts.

The organization of the paper is as follows. In section 2, we briefly explain the concept learning problem and the PAC framework. In section 3, we describe our algorithm and analyze its convergence properties. We show in section 4 how the algorithm can be modified to learn a class of disjunctive concepts. We give a comparison of our algorithm with other techniques in section 5. We give empirical results of our algorithm on a few synthetic and real–world domains in section 6 and 7 contains some concluding remarks.

## 2. Concept Learning

As mentioned in the previous section, the problem domain is specified by a finite set of attributes. For our purposes, the attributes are distinguished into two types based on the structure imposed on the set of values possible for the attribute. If the value set of an attribute has no structure, then the attribute is called nominal; if the value set is totally ordered, then it is called linear. In this paper, all linear attributes are assumed to be real-valued.

Let the attributes(nominal or linear) chosen for the domain be $Y_i$ that take values from sets $V_i$, $i = 1,..., N$.

*Definition* 1. *A concept description* (*logic expression*) *given by*

$$[Y_1 \in v_1] \Lambda ... \Lambda [Y_N \in v_N]$$

*where $v_i$ is a subset of $V_i$, $i = 1,\ldots, N$, and further if $Y_i$ is a linear attribute then $v_i$ is an interval, is said to be* `simple conjunctive expression.`

Here a term of the form $[Y_i \in v_i]$ is an indicator of whether the attribute $Y_i$ belongs to the set $v_i$. This is sometimes referred to as internal disjunction because it can be viewed as a disjunction of equality predicates.

*Remark* 2.1. *It can be noted that a collection of subsets $(v_1,\ldots, v_N)$, where $v_i \subset V_i$, uniquely defines a simple conjunctive concept. Therefore, we denote the conjunctive concept by $(v_1,\ldots, v_N)$.*

*Definition* 2. *A concept description of the form*

$$C_1 \vee C_2 \vee \ldots \vee C_k$$

where each $C_i$ is a simple conjunctive expression is said to be a *k*-`term disjunctive expression.`

We use a simple conjunctive expression or a *k*-term disjunctive expression to represent our concepts. The choice of a particular representation defines the so-called hypothesis space, a space of all concepts expressible in the chosen representation. We formulate the concept learning problem as a search over the hypothesis space for the "best" concept. Before defining what we mean by the "best" concept, we discuss some of the issues involved in the learning problem.

In a real-life problem, the space of examples may be so large that the teacher-classified examples supplied to the learning system may form only a small subset of this space. We expect the algorithm to learn a concept which performs well not just on these examples but also on the unseen objects of the domain. This issue of generalization is intimately related to how representative are the examples given to the learning system. This can be formulated precisely through the PAC framework as explained in the next subsection. Also, the teacher may not be infallible and consequently the pre-classified examples may be noisy. Thus, sometimes the learning system might even receive two examples having the same values for the attributes but one classified as positive and the other negative. Noisy examples may be due to mistakes in teacher classification, errors in attribute measurements or representational inadequacy (e.g. overlapping class conditional densities in a pattern classification problem). The objective of learning has to be defined to handle both the issues of generalization and noise. With this motivation, we adopt the PAC Learning framework[10–12] described below.

## 2.1. *Probably approximately correct learning*

Consider a learning system interacting with a teacher. The teacher presents the learner with randomly drawn examples, each example consisting of an instance $x \in X$ and an outcome $y \in Y$, where $X$ and $Y$ are called *instance* and *outcome* spaces respectively. These examples are generated independently according to a joint probability distribution, say P, on $X \times Y$, unknown to the learner. Using these so-called training examples, the learner chooses a hypothesis from the *hypothesis* space $H$ based on a learning algorithm $\mathcal{A}$. A hy-

pothesis $h$ belonging to $H$ has domain $X$ and range $A$ known as the *decision* space. ($X$, $Y$ and $A$ are arbitrary sets.) The hypothesis $h$ chosen by the learner is evaluated through a function $l$ called the *loss* function. The loss function is a fixed real-valued function defined on $Y \times A$ and is assumed to be known to the learner. Given an example $(x, y)$, this function measures the loss suffered by the learner for choosing the *action* $h(x)$ against the outcome $y$. The objective of the learner is to choose a hypothesis that minimizes the expected loss. Formally, define

$$L(h) = E[l(y, h(x))] \tag{2.1}$$

$$h^* = arg \min_h L(h) \tag{2.2}$$

where the expectation is w.r.t $P$ and $h^*$ is the concept having the minimal expected loss and is called the *correct* concept.

The following definition characterizes the objective of the learner. It can be noted that the definition is a special case of the general definition given by Haussler[12].

*Definition* 3. *Let $X$, $Y$, $P$ be as above and $\mathcal{A}$ be the learning algorithm used by the learner. Let $h_n \in H$ be the hypothesis output by the algorithm after $n$ training examples are presented by the teacher. Then, the algorithm* PAC learns *if, for any $h^*$, $L(h_n)$ converges to $L(h^*)$ (see Remark 2.2 below) as $n$ goes to infinity, irrespective of the distribution $P$.*

*Remark* 2.2. *In the above definition, the convergence of $L(h_n)$ to $L(h^*)$ has to be defined stochastically since $L(h_n)$ is a random variable. In COLT literature, this convergence is usually assumed to be in probability*[11,12]*. In this paper, we use weak convergence for this purpose. However, as in COLT, we demand the convergence to be uniform over all probability distributions $P$ and the target concept class.*

Remark 2.3. *In PAC framework, the generalization issue is handled by presenting the examples drawn independently from a fixed but arbitrary distribution $P$ and measuring the error (or loss) of the learnt concept with respect to the same probability distribution $P$. The examples, being independent and identically distributed (i.i.d.), cannot form a cleverly chosen sequence and hence become representative of the target concept.*

If the loss function is such that $l(y, a)$ is 1 when $y \neq a$ and is 0 otherwise, then the correct concept $h^*$ as given by (2.2) will be the concept having minimum probability of misclassification as used in Pattern recognition[2,14]. Formulating the problem using this loss function, our aim will be to develop an algorithm that PAC learns in the sense of Definition 3. That is, the algorithm learns the concept having minimum probability of misclassification, asymptotically as the number of examples goes to infinity, given an i.i.d sequence of examples that are drawn w.r.t any distribution.

### *Problem definition*

For us, the concept learning problem is specified by the following.

- A finite set of characteristic attributes (nominal or linear) of the domain are specified. Each nominal attribute assumes only finitely many values and each linear at-

tribute takes values from a bounded interval in $\mathcal{R}$. All examples will be described as tuples of values for these attributes.

- The right concept can be described as a logic expression involving a subset of these attributes.

The choice of attributes defines the instance space of our problem. Since we are considering a concept learning problem, the outcome space is {0, 1}. Here, we assume that a negative example is indicated by the label 0 and a positive example by 1. The decision space is also {0, 1}. The hypothesis space is the class of all simple conjunctive expressions given by Definition 1. We choose the loss function $l(y, a) = I\{y \neq a\}$[§]. Hence, the correct concept $h^*$ is the one having minimum probability of misclassification.

In the next section we present an algorithm based on a cooperating team of learning automata for the concept learning problem. Teams of automata have been used earlier for concept learning[15,13]. These models can learn concepts with nominal attributes only and thus cannot tackle *true* linear attributes. The model proposed in this paper alleviates this problem and can be considered as a generalization of Sastry *et al*[13]. to handle both nominal and linear attributes. We also show how the algorithm can be used for learning some disjunctive concepts.

Since we use the class of simple conjunctive expressions as our hypothesis space, we give some more definitions useful for the conjunctive concepts.

*Definition* 4. *Let* $(v_1^*,\ldots,v_N^*)$ *be the correct conjunctive concept (defined by equation* (2.2)). *Then* $v_1^*$ *is called the* correct set *of the attribute* $Y_i$, $i = 1,\ldots, N$.

It is easy to see that the value of any attribute in a positive example belongs to the correct set of that attribute and in any negative example the value of at least one attribute does not belong to the correct set. Also it should be noted that this notion of correct set is meaningful only for simple conjunctive concepts (cf. Definition 1). The problem of learning disjunctive concepts with our method is addressed in Section 4.

## 3. A stochastic algorithm for learning simple conjunctive expressions

In this section we first introduce the concept of a cooperative game of Learning Automata. Then we show how the model can be used for learning simple conjunctive concepts in a noisy environment. Our treatment of learning automata will be brief. It is introduced only to explain our notation and state a theorem needed to prove the correctness of our algorithm. The reader is referred to Narendra and Thathachar[16] and Santharam[17] for more details.

### 3.1. *Learning automata*

Learning automata are adaptive decision making devices that learn the optimal action from the available set of actions by interacting with a random environment. The learning automaton can be classified based on its action set into Finite Action set Learning Automaton (FALA) and Continuous Action set Learning Automaton (CALA).

§ $I\{A\}$ denotes the indicator function of set $A$

### 3.1.1. *Finite Action set Learning Automata*

A FALA has a finite set of actions $A = \{a_1, a_2,..., a_r\}$ from which it chooses an action at each instant. This choice is made randomly based on the so-called action probability distribution. Let $\mathbf{p}(k) = [p_1(k), p_2(k),..., p_r(k)]$ denote this distribution, where $p_i(k)$ is the probability of choosing action $a_i$ at instant $k$, $k = 0, 1, 2,....$ For the choice of action, the automaton gets a reaction(which is called response or reinforcement) from the environment. This reaction is stochastic whose expected value is $d_i$ if the automaton has chosen $a_i$. The values $d_i$, $i = 1,..., r$, are called the reward probabilities of the environment and are unknown to the automaton. The objective of the automaton is to maximize the expected value of reinforcement. Let $d_m = \max_i d_i$. Then $a_m$ is called the optimal action and the aim is to identify this optimal action; that is, to evolve into a state where in the optimal action is chosen with a probability arbitrarily close to unity. This is to be achieved through a learning algorithm that updates, at each instant $k$, the action probability distribution $\mathbf{p}(k)$ into $\mathbf{p}(k+1)$ using the most recent interaction with the environment, namely, the pair $(\alpha(k), \beta(k))$. Thus if $T$ represents the learning algorithm, then, $\mathbf{p}(k+1) = T(\mathbf{p}(k), \alpha(k), \beta(k))$. The main problem of interest here is the design of learning algorithms with satisfactory asymptotic behaviour. We are interested in algorithms that make $p_m(k)$ converge to a value close to unity in some sense.

*Definition* 5. *A learning algorithm is said to be* $\epsilon$-optimal *if given any* $\epsilon > 0$, we can choose parameters of the learning algorithm such that with probability greater than $1 - \epsilon$,

$$\liminf_k \; p_m(k) > 1 - \epsilon.$$

A learning algorithm for FALA that we use later on, called the Linear Reward-Inaction ($L_{R-I}$) algorithm[16] is described below.

Let $a(k) = a_i$ and let $r(k)$ be the response obtained at $k$. Then $\mathbf{p}(k)$ is updated as follows:

$$\begin{aligned} p_i(k+1) &= p_i(k) + \lambda r(k)[1 - p_i(k)] \\ p_j(k+1) &= p_j(k) - \lambda r(k) p_j(k), \; \forall j \neq i \end{aligned} \tag{3.1}$$

where $\lambda \in (0, 1)$ is a parameter of the algorithm.

The $L_{R-I}$ algorithm is known to be $\epsilon$-optimal under stationary environments[16].

### 3.1.2. *Continuous Action set Learning Automata*

The CALA is similar to an FALA with the main difference being that the action set now is continuous. The CALA chooses actions from the real line $\mathcal{R}$. The action probability distribution at $k$ is $N(\mu(k), \sigma(k))$, the normal distribution with mean $\mu(k)$ and standard deviation $\sigma(k)$. At each instant, the CALA updates its action probability distribution (based on its interaction with the environment) by updating $\mu(k)$ and $\sigma(k)$, which is analogous to updating the action probabilities by the FALA. Since the action set is continuous, instead of reward probabilities for various actions, we now have a reward probability function, $f: \mathcal{R} \to \mathcal{R}$, defined by

$$f(x) = \mathrm{E}[\text{Reinforcement} \mid x \text{ is the action chosen}] \tag{3.2}$$

Like in the case of reward probabilities $d_i$, for CALA, the reward probability function $f$ is unknown to the automaton.

We shall denote the reinforcement in response to action $x$ as $r_x$ and thus

$$f(x) = Er_x.$$

The objective for CALA is to learn the value of $x$ at which $f$ attains a maximum. That is, we want the action probability distribution, $N(\mu(k), \sigma(k))$ to converge to $N(x_o, 0)$ where $x_o$ is a maximum of $f$. However, for some technical reasons we cannot let $\sigma(k)$ to converge to zero. So, we use another parameter, $\sigma_L > 0$, and keep the objective of learning as $\sigma(k)$ converging close to $\sigma_L$ and $\mu(k)$ converging to a maximum of $f$. By choosing $\sigma_L$ sufficiently small, asymptotically CALA will choose actions sufficiently close to the maximum with a probability sufficiently close to unity[17].

The learning algorithm for CALA is described below. Since the updating given for $\sigma(k)$ does not automatically guarantee that $\sigma(k+1) \geq \sigma_L$, we always use a projected version of $\sigma(k)$, denoted by $\phi(\sigma(k))$, while choosing actions. Also, unlike FALA, CALA interacts with the environment through a choice of two actions at each instant.

At each instant $k$, the CALA chooses $x(k) \in \mathcal{R}$ at random from its current distribution $N(\mu(k), \phi(\sigma(k)))$ where $\phi$ is the function specified below. Then it gets the reinforcement from the environment for the two actions: $\mu(k)$ and $x(k)$. Let these reinforcements be $r_\mu$ and $r_x$. Then, the action probability distribution is updated as

$$\begin{aligned} \mu(k+1) &= \mu(k) + \lambda F_1(\mu(k), \sigma(k), x(k), r_x(k), r_\mu(k)) \\ \sigma(k+1) &= \sigma(k) + \lambda F_2(\mu(k), \sigma(k), x(k), r_x(k), r_\mu(k)) - C[\sigma(k) - \sigma_L] \end{aligned} \tag{3.3}$$

where $F_1(\cdot)$, $F_2(\cdot)$ and $\phi(\cdot)$ are defined as below:

$$F_1(\mu, \sigma, x, r_x, r_\mu) = \left(\frac{r_x - r_\mu}{\phi(\sigma)}\right)\left(\frac{x-\mu}{\phi(\sigma)}\right)$$

$$F_2(\mu, \sigma, x, r_x, r_\mu) = \left(\frac{r_x - r_\mu}{\phi(\sigma)}\right)\left[\left(\frac{x-\mu}{\phi(\sigma)}\right)^2 - 1\right]$$

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

and $\sigma_L, C > 0, \lambda \in (0, 1)$ are parameters of the algorithm.

For this algorithm it is proved that with arbitrarily large probability, $\mu(k)$ will converge close to a maximum of $f(\cdot)$ and $\sigma(k)$ will converge close to $\sigma_L$, if we choose $\lambda$ and $\sigma_L$ sufficiently small and $C$ sufficiently large[17].

### 3.1.3. *Games of Automata*

From the previous section it is clear that the automaton can be used for finding the optimum value of a parameter, especially when only noise-corrupted values of the criterion function being optimized are available. This is done by making the set of actions the same

as the set of possible values of the parameters and the reinforcement signal as the noise corrupted value of the function to be optimized at the chosen parameter value. (The parameter range may have to be discretized if we use a FALA). A single automaton can learn the optimum value of a single parameter and hence it will suffice for unidimensional problems. But for multidimensional problems, we need to consider a system of several automata and in this context we consider the case where these automata are involved in a game with common payoff. In such a game, the automata correspond to the players of the game and its actions to the various strategies available to the players. Consider a team of learning automata consisting of $N$ Finite action set learning automata and $M$ Continuous actions set learning automata. Let the action set of $i$-th FALA be denoted by $S_i$ with $|S_i| = n_i$, $1 \le i \le N$. All the CALA choose actions from the real line $\mathcal{R}$. Let $a_i(k) \in S_i$, $i = 1,\ldots, N$, denote the action chosen by the $i$-th FALA and $x_j(k) \in \mathcal{R}$, $j = 1,\ldots, M$, the action chosen by the $j$-th CALA at the $k$-th instant. $\mathbf{a}(k) = (a_1(k), a_2(k),\ldots, a_N(k))$ represents the actions chosen by the FALA part of the team and $\mathbf{x}(k) = (x_1(k), x_2(k),\ldots, x_M(k))$, the set of actions chosen by the CALA part of the team at instant $k$. Define $q(k) = (\mathbf{a}(k), \mathbf{x}(k)) \in \left(\prod_{i=1}^{N} S_i\right) \times \mathcal{R}^M$. $q$ collectively denotes the actions chosen by the team. At each instant, the team interacts twice with the environment. Let $r_{\mathbf{x}}(k)$ and $r_{\mu}(k)$ be the payoffs obtained during the two interactions at the $k$-th instant. $r_{\mathbf{x}}(k)$ denotes the payoff obtained when the FALA part of the team chooses $\mathbf{a}(k)$ and the CALA part of the team chooses $\mathbf{x}(k)$ and $r_{\mu}(k)$ denotes the payoff obtained when the FALA part chooses $\mathbf{a}(k)$ and the CALA part chooses $\mu(k)$. The payoffs are given as common reinforcement to all the automata. Using these payoffs, the team updates its state, chooses actions at the next instant and the cycle repeats itself. Define, for $q = (\mathbf{a}, \mathbf{x})$, $\mathbf{a} = (a_1,\ldots, a_N)$, $\mathbf{x} = (x_1,\ldots, x_M)$,

$$g(q) = \mathrm{E}[\text{Reinforcement} \mid i\text{-th FALA chose } a_i \in S_i \text{ and } j\text{-th CALA chose } x_j \in \mathcal{R}]. \quad (3.4)$$

$g(\cdot)$, called the payoff function, captures the reward structure of the game and is unknown to the automaton. The goal of the team is to choose actions to globally maximize the payoff function.

*Definition* 6. *Let* $q^* = (\mathbf{a}^*, \mathbf{x}^*)$, $\mathbf{a}^* = \left(a_1^*,\ldots, a_N^*\right); \mathbf{x}^* = \left(x_1^*,\ldots, x_M^*\right)$.. *We say* $q^*$ *is an optimal point of* $g(\cdot)$ *if*

1.

$$g(q^*) \ge g((\mathbf{a}, \mathbf{x}^*)),\ \forall \mathbf{a} = (a_1,\ldots, a_N) \text{ s.t. } \exists j \text{ s.t. } a_i = a_i^*,\ \forall i \ne j \text{ and } a_j \ne a_i^*.$$

2. $\exists \in > 0$ *such that*

$$g(q^*) \ge g((\mathbf{a}^*, \mathbf{x})),\ \forall \mathbf{x} \in \mathcal{B}^M(\mathbf{x}^*, \in)$$

*where* $\mathcal{B}^M(\mathbf{x}^*, \in)$ *denotes the open ball in* $R^M$ *of radius* $\in$ *and centre* $\mathbf{x}^*$.

*Remark* 3.1. *In Definition* 6, *condition* 1 *implies that* $a^*$ *is a mode when the payoff function* $g((\mathbf{a}, \mathbf{x}))$ *is thought of as an N dimensional matrix indexed by* $a_i$, $I = 1,\ldots, N$, *by fixing*

*x. Condition* 2 *means that* $\mathbf{x}^*$ *is a local maximum of the payoff function* $g((\mathbf{a}, \mathbf{x}))$ *when considered as a function of* $\mathbf{x}$ *only.*

The theorem given below characterizes the convergence behaviour of the game played by the hybrid team of FALA and CALA.

*Theorem* 1. *Let* $g(\cdot)$ *be the payoff function of a cooperative game with common payoff played by N FALA and M CALA. Let all the FALA use identical* $L_{R-I}$ *algorithms and all CALA use the algorithm given by* (3.3.). *Then, asymptotically as* $\lambda \to 0$, *the team converges to one of the optimal points of the payoff function* $g(\cdot)$.

The theorem for the special case of the team consisting only of FALA is proved in Sastry *et al*[18]. and for the case of team consisting only of CALA in Santaram[17]. The proof for the general case of a hybrid team consisting of FALA and CALA is a simple extension and is available in Rajaraman *et al.*[19].

### 3.2. *Algorithm for concept learning*

We formulate the problem of concept learning as a cooperative game with a common payoff played by a team of FALA and CALA. For each nominal attribute, the correct set is a subset with finite cardinality. Therefore, to model a nominal attribute, we may use a FALA having as many actions as the number of possible subsets. Since this requires an exponential number of actions, we learn the correct set of a nominal attribute by employing a two action FALA for each value of that attribute to decide whether or not the value belongs to the correct set. For each linear attribute, we use two CALA to learn the end points of the interval which forms the correct set of that attribute.

Our notation is as follows.

$Y_i^{(d)}$ the $i$-th nominal attribute, $i = 1, \ldots, N$.

$V_i^{(d)}$ the value set of $i$-th nominal attribute $Y_i^{(d)}$. It equals $\{x_{i1}, \ldots, x_{in_i}\}$.

$X_{ij}^{(d)}$ the automaton(FALA) representing $j$-th value of $i$-th nominal attribute(*i.e.* $x_{ij}$), $j = 1, \ldots, n_i$; $i = 1, \ldots, N$.

$\alpha_{ij}$ the action chosen by the automaton $X_{ij}^{(d)}$. It belongs to $\{YES, NO\}$

$Y_{N+i}^{(c)}$ the $i$-th linear attribute, $i = 1, \ldots, M$.

$V_i^{(c)}$ the value set of $i$-th linear attribute $Y_i^{(c)}$. It is assumed to be an interval in $\mathcal{R}$.

$X_{il}^{(c)}$ the automaton(CALA) representing one end point of the correct set of $i$-th linear attribute, $l = 1, 2$; $i = N + 1, \ldots, N + M$. $l = 1(2)$ denotes the left(right) end point.

$w_{il}$ the action chosen by the automaton $X_{il}^{(c)}$

$v_i$ a subset of either $V_i^{(d)}$ or $V_i^{(c)}$ depending on whether $i$-th attribute is nominal or linear respectively.

Our model consists of a team of $n_1 + n_2 + ...+ n_N$ FALA, $X_{ij}^{(d)}$ ($j = 1,..., n_i$; $i = 1,..., N$), and $2M$ CALA, $X_{il}^{(c)}$($l = 1, 2$; $i = N + 1,..., N + M$), involved in a game with common payoff. The FALA part of the team is concerned with choosing the correct sets for nominal attributes. Since we use one FALA to represent each value of a nominal attribute, $n_1 + n_2 + ...+ n_N$ automata are needed to represent the $N$ nominal attributes. The FALA $X_{ij}^{(d)}$ is concerned with the decision of whether $x_{ij}$ is in the correct set of $Y_{ij}^{(d)}$. Each FALA has two actions YES and NO. Let $p_{ij}(k)$ denote the probability with which $X_{ij}^{(d)}$, will choose action YES at instant $k$. Since each FALA has only two actions, $p_{ij}(k)$ completely defines the action probability distribution for $X_{ij}^{(d)}$. The CALA part of the team learns the correct sets for the linear attributes. Since we assumed that the correct sets of linear attributes can be expressed as intervals, we need two CALA for every linear attribute, each automaton learning one end point. The CALA, $X_{il}^{(c)}$, is concerned with the choice of one end point($l = 1, 2$ respectively denote the left and right end points) of the $i$-th linear attribute. Let $\mu_{il}(k)$ and $\sigma_{il}(k)$ denote the mean and variance of the normal distribution for $X_{il}^{(c)}$ at instant $k$. Hence, the normal distribution $N(\mu_{il}(k), \phi(\sigma_{il}(k)))$ is the action probability distribution for $X_{il}^{(d)}$ at $k$-th instant, where $\phi$ is the function defined in section 3.1.2 to project the variance. The team functions as follows.

As explained in section 3.1.2, a CALA needs to interact with the environment twice at each instant. Hence the team consisting of a number of FALA and CALA also interacts with environment twice at each instant.

At every instant $k$, each of the automata chooses an action(YES or NO for FALA and a real number for CALA) independently and at random based on its current action probability distribution. As per our notation, let $\alpha_{ij}$ denote the action chosen by $X_{ij}^{(d)}$, $j = 1,..., n_i$; $i = 1,..., N$, and $w_{il}$ denote the action chosen by $X_{il}^{(c)}$, $l = 1, 2$; $i = N + 1,..., N + M$. Now consider the conjunctive concepts: $C = (v_1,..., v_{N+M})$ and $C' = (v_1,..., v_N, v'_{N+1},..., v'_{N+M})$,, defined by

$$v_i = \begin{cases} \{x_{ij} | \alpha_{ij} = YES,\ j = 1,..., n_i\} & i = 1,..., N \\ [w_{i1}, w_{i2}] & i = N+1,..., N+M \end{cases}$$

$$v'_{N+i} = [\mu_{i1}, \mu_{i2}],\ i = 1,..., M.$$

Then, the responses are generated for the team by classifying the next example with concepts $C$ and $C'$ respectively. The response is 1 if the classification agrees with that by the teacher and 0 otherwise(See equation (3.5) below). The automata update the action probability distributions using the learning algorithms specified in sections 3.1.1. and 3.1.2. If the action probability vectors of all FALA converge to unit vectors* and the

*Theoretically, we desire all the probabilities attain the value unity to ascertain convergence. However, as our algorithm achieves only asymptotic convergence, we verify the convergence if all the probabilities go above a value close to unity, say 0.99.

means and variances of all CALA converge then we say that the team has converged to the concept $(v_1,\ldots, v_{N+M})$ where

$$v_i = \begin{cases} \{x_{ij} | p_{ij} = 1, j = 1,\ldots,n_i\} & i = 1,\ldots,N \\ [\mu_{i1}, \mu_{i2}] & i = N+1,\ldots,N+M \end{cases}$$

The complete algorithm is given below.

**Algorithm-1**

Initially, we set $p_{ij}(0) = 1/2$, $\forall j = 1,\ldots, n_i$; $i = 1,\ldots, N$. $\mu_{il}(0)$, $i = N+1,\ldots, N+M$, is chosen to be some real number. (If we know that the $i$-th linear attribute has range $[C_{i1}, C_{i2}]$, then we can choose $\mu_{i1}(0) = C_{i1}$ and $\mu_{i2}(0) = C_{i2}$.) $\sigma_{il}(0)$ is set to a suitably large positive value. At every instant, each of the automata $X_{ij}^{(d)}$ and $X_{il}^{(c)}$ simultaneously and independently chooses actions at random based on its current action probability distribution and the team interacts with the environment through two sets of actions as explained above.

The responses to the team are

$$r = \begin{cases} 1 & \text{if the classification by concept } C \text{ matches with teacher's classification} \\ 0 & \text{otherwise} \end{cases}$$
$$r' = \begin{cases} 1 & \text{if the classification by concept } C' \text{ matches with teacher's classification} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

Then, each FALA $X_{ij}^{(d)}$ updates $p_{ij}$ as follows:

$$p_{ij}(k+1) = \begin{cases} p_{ij}(k) + \lambda r'(k)[1 - p_{ij}(k)] & \text{if } \alpha_{ij}(k) = \text{YES} \\ p_{ij}(k) - \lambda r'(k) p_{ij}(k) & \text{if } \alpha_{ij}(k) = \text{No} \end{cases}$$

where $\lambda \in (0, 1)$ is a parameter of the algorithm.

Each CALA $X_{il}^{(c)}$ updates $\mu_{il}$ and $\sigma_{il}$ as follows:

$$\mu_{il}(k+1) = \mu_{il}(k) + \lambda F_1^{il}(\mu_{il}(k), \sigma_{il}(k), w_{il}(k), r(k), r'(k)) \tag{3.6}$$
$$\sigma_{il}(k+1) = \sigma_{il}(k) + \lambda F_2^{il}(\mu_{il}(k), \sigma_{il}(k), w_{il}(k), r(k), r'(k)) - C[\sigma_{il}(k) - \sigma_L].$$

where $F_1^{il}$, $F_2^{il}$ are defined as below:

$$F_1^{il}(\mu, \sigma, x, r, r') = \left(\frac{r - r'}{\phi(\sigma)}\right)\left(\frac{x - \mu}{\phi(\sigma)}\right)$$

$$F_2^{il}(\mu, \sigma, x, r, r') = \left(\frac{r - r'}{\phi(\sigma)}\right)\left[\left(\frac{x - \mu}{\phi(\sigma)}\right)^2 - 1\right]$$

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

and $C > 0$, $\lambda \in (0, 1)$ are parameters of the algorithm.

The next subsection presents the convergence analysis of Algorithm-1. Before proceeding with the analysis it may be noted that our formulation of the concept learning problem includes possibility of noisy samples or a noisy teacher. The response to the automata team is determined by whether or not the classification by the team agrees with that of the teacher and not on whether the classification of the example is *correct*.

Recall from Section 2.1 that $P$ is the probability distribution defined over the space of classified examples $X \times \{0, 1\}$ for our problem(see Section 2.2). By this definition, noise in the examples can be taken care of by assigning non-zero probabilities to examples described by the same instance but having different class labels. The correct concept defined by equation (2.2) is then the concept having minimum expected classification error w.r.t $P$, i.e. the concept having minimum probability of misclassification. Supposing that this error probability is $\eta\%$, we can equivalently model this situation by defining on the instance space another distribution $P'$, which is the marginal distribution of $P$ over $X$, and assuming that $\eta\%$ of the examples in $X$ are misclassified. Since the examples are i.i.d in PAC formulation, this noise model is implied by the model where every example is classified correctly(i.e with the 'correct' concept) and then the class label corrupted with $\eta\%$ probability in an unbiased manner before being presented to the learning system. The only difference between this model and that of an arbitrary distribution $P$ over $X \times \{0, 1\}$ from which examples are drawn i.i.d. is the assumption of unbiasedness in the 'corruption' of class label(with respect to that of the concept with minimum error) that an example undergoes. In the following subsection, Algorithm-1 is analyzed within this noise model. We compare our noise model with other models in Section 5.

### 3.2.1. *Analysis of Algorithm-1*

The actions chosen by the hybrid team of automata will be a tuple of $n_1 + n_2 + \ldots + n_N + 2M$ elements. Using the notation of section 3.1.3, we denote this action tuple by $q = (\alpha, w)$ where $\alpha = (\alpha_{ij}, j = 1,\ldots, n_i;\ i = 1,\ldots, N)$, with $\alpha_{ij}$ being the action chosen by the FALA representing the $j$-th value of the $i$-th nominal attribute (*i.e.* $x_{ij}$), and $w = (w_{il}, l = 1, 2;\ i = 1,\ldots, M)$, with $w_{il}$ being the action chosen by the CALA learning one end point(left or right depending on whether $l = 1$ or 2 respectively) of the $i$-th linear attribute. As noted in section 3.2, the action tuple $q$ corresponds to the conjunctive concept $(v_1,\ldots, v_{N+M})$ where

$$v_i = \begin{cases} \{x_{ij} | \alpha_{ij} = YES, j = 1,\ldots, n_i\} & i = 1,\ldots, N \\ [w_{i1}, w_{i2}] & i = N+1,\ldots, N+M \end{cases} \tag{3.7}$$

Suppose $g_\eta(\cdot)$ is the payoff function, defined by (3.4), for the team of automata with the reward structure as in (3.5), when the teacher classifies examples with $\eta\%$ noise. Let $g(\cdot)$ be the payoff function under no noise. In view of Theorem 1, we know that the team converges to one of the optimal points of the payoff function $g_\eta(\cdot)$ and the goal of the

analysis is to characterize these optimal points. The analysis will follow in two steps. First, we consider the effect of noise on the payoff function. In Lemma 1, we show that the optimal points of $g_\eta(\cdot)$ are identical to those of $g(\cdot)$, if $\eta < 50$. Thus, under the condition that $\eta < 50$ it is enough to analyze the optimal points of $g(\cdot)$. We analyze the optimal points of $g(\cdot)$ in Theorem 2, thus proving the convergence of our algorithm.

*Lemma* 1. *Let $g_\eta(\cdot)$ be the payoff function under $\eta$% classification noise and $g(\cdot)$ be the payoff function under no noise. Then, the optimal points of $g_\eta(\cdot)$ and $g(\cdot)$ are identical if*

1. *the teacher is unbiased*
2. $\eta < 50$

*Proof*

The expected reward to the team, under $\eta$% noise, for a set of actions chosen by the automata in the team resulting in subsets $v_i$, $I = 1,\ldots, N + M$, of the range of the attributes $Y_i^{(d)}, i = 1,\ldots, N$ and $Y_i^{(c)}, i = 1,\ldots, M$ is given by

$$
\begin{aligned}
&g_\eta(v_1,\ldots, v_{N+M})\\
&\quad = p_+ \cdot Pr[\text{On a random positive example}\\
&\qquad \text{the teacher's classification matches with that by the team}\\
&\quad + p_- \cdot Pr[\text{On a random negative example}\\
&\qquad \text{the teacher's classification matches with that by the team}]
\end{aligned}
$$

where $p_+ = Pr$[A random example is positive] and $p_- = Pr$[A random example is negative].

Since the teacher is unbiased,

$$
\begin{aligned}
&Pr[\text{a random positive example is classified negative by the teacher}]\\
&\quad = Pr[\text{a random negative example is classified positive by the teacher}]\\
&\quad = \frac{\eta}{100}
\end{aligned}
$$

and

$$
\begin{aligned}
&Pr[\text{a random positive example is classified positive by the teacher}]\\
&\quad = Pr[\text{a random negative example is classified negative by the teacher}]\\
&\quad = 1 - \frac{\eta}{100}
\end{aligned}
$$

Therefore,

$$g_\eta(v_1,\ldots,v_{N+M}) = \left(1 - \frac{\eta}{50}\right) g(v_1,\ldots,v_{N+M}) + \frac{\eta}{100}$$

The optimal points remain unchanged as in non-noisy environments if

$$1 - \frac{\eta}{50} > 0$$

i.e. if $\eta < 50$. Thus, the lemma follows.

Before stating Theorem 2, we introduce some notation useful in proving Theorem 2. Let $\mathcal{D} = \left(\prod_{i=1}^{N} V_i^{(d)}\right) \times \left(\prod_{i=1}^{M} V_{N+i}^{(c)}\right)$. Let the preclassified examples given by the teacher be drawn from a distribution over $\mathcal{D} \times \{0, 1\}$ and let $\rho$ be the underlying probability measure. We assume that $\rho$ satisfies the following.

**(A-1)**

$$\rho(A \times \{0, 1\}) > 0, \forall A = v_1 \times v_2 \times \ldots \times v_{N+M}, A \subset \mathcal{D},$$

where $v_i \subset V_i^{(d)}$, $v_i \neq \phi$, $I = 1, \ldots, N$ and $v_i = [w_{i1}, w_{i2}] \subset V_i^{(c)}$ such that $w_{i1} < w_{i2}$ and $w_{il} \in \mathcal{R}$, $l = 1, 2$; $i = N + 1, \ldots, N + M$.

*Remark* 3.2. *Assumption* (A-1) *states that instances belonging to sets A of the form as in* (A-1) *have non-zero probability of being selected by the teacher. Hence, by this assumption, constructing concepts by adding* (*or deleting*) *such sets A to* (*or from*) *a concept will result in a strict increase or decrease in the expected loss as given by* (2.1). *This fact will be useful in proving whether the given concept is an optimal point or not, in Theorem* 2 *below.*

*Theorem* 2. *Consider the automata team involved in a game with common payoff as described in section* 3.2 *for a concept learning problem that satisfies assumption* (A-1). *Assume that there is no classification noise. Then, the following characterizes the optimal points of the payoff function.*

1. *the tuple* $\left(v_1^*, \ldots, v_{N+M}^*\right)$ *where each* $v_i^*$ *is the correct set* (*cf. Definition* 4) *for the i-th attribute, is an optimal point.*
2. *In all other optimal points* $(v_1, \ldots, v_{N+M})$, *at least one of the* $v_i$*-s is a null set.*

*Proof*

$$\begin{aligned} & g(v_1, \ldots, v_{N+M}) \\ & \quad = p_+ \cdot Pr[a \text{ random positive example is classified positive by } (v_1, \ldots, v_{N+M})] \\ & \qquad + p_- \cdot Pr[\text{a random negative example is classified negative by } (v_1, \ldots, v_{N+M})] \\ & \quad = p_+ P^+(v_1, \ldots, v_{N+M}) + p_- P^-(v_1, \ldots, v_{N+M}) \end{aligned}$$

where

$$\begin{aligned} & P^+(v_1, \ldots, v_{N+M}) \\ & \quad = \rho[\{(x_1, \ldots, x_{N+M}, 1) | x_i \in v_i, i = 1, \ldots, N + M\}] \\ & P^-(v_1, \ldots, v_{N+M}) \\ & \quad = \rho[\{(x_1, \ldots, x_{N+M}, 0) | x_i \notin v_i, \text{ for at least one } i\}] \end{aligned}$$

with $\rho[\cdot]$ being the underlying probability measure as in (A-1)

It is easy to see that $P^+$ and $P^-$ satisfy the following monotone properties because of assumption A-1.

Let $v_i' \subset v_i, i = 1,\ldots,N+M$. Then,

(M-1) For all $i$,

$$P^+(v_1,\ldots, v_{N+M}) \geq P^+(v_1,\ldots, v_{i-1}, v_i', v_{i+1},\ldots, v_{N+M})$$

where $v_i' \subset v_i$.. The inequality is strict iff $(v_i - v_i') \bigcap v_i^* \neq \phi$ and $P^+(v_1,\ldots, v_{N+M}) > 0$

(M-2) For all $i$,

$$P^-(v_1,\ldots, v_{N+M}) \leq P^-(v_1,\ldots, v_{i-1}, v_i', v_{i+1},\ldots, v_{N+M})$$

where $v_i' \subset v_i$. The inequality is strict iff $(v_i - v_i') \bigcap (v_i^*)^c \neq \phi$ and $P^-(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_{N+M}) > 0$.

Let $v_i^*$ be the correct set for the $i$-th attribute.

To prove the theorem, we have to show

- $(v_1^*,\ldots,v_{N+M}^*)$ is an optimal point
- $(v_1,\ldots, v_{N+M})$ is not an optimal point when no $v_i$ is null and at least one $v_i$ does not equal $v_i^*$

*Part* I

We show here that $(v_1^*,\ldots,v_{N+M}^*)$ is an optimal point. To show this, we have to prove the following.

- The payoff function does not increase by choosing a subset, for any nominal attribute, that differs from the correct set of the respective attribute in exactly one value.
- The payoff function does not increase by choosing a subset, for any linear attribute, which is identical to the correct set of the respective attribute except for exactly one end point that is atmost $\in$ distance from the corresponding end point in the correct set, for all sufficiently small $\in$. (It may be recalled that for a linear attribute the correct set is a single interval.)

That is, we have to prove

1.

$$g(v_1^*,\ldots,v_{N+M}^*) \geq g(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*), \quad \forall i = 1,\ldots,N$$

such that $v_i^*$ and $v_i'$ differ in exactly one value.

2. $\exists \in^* > 0$ *s.t.* $\forall 0 < \in < \in^*$,

$$g(v_1^*,\ldots,v_{N+M}^*) \geq g(v_1^*,\ldots,v_{i-1}^*,v_i'(\in),v_{i+1}^*,\ldots,v_{N+M}^*), \quad \forall i = N+1,\ldots,N+M$$

where $v_i^*$ and $v_i'(\epsilon)$ are intervals with one end point common and the other end points differing by atmost $\epsilon$.

*Case* 1

Since $v_i'$ and $v_i^*$ differ in exactly one value, either $v_i'$ contains an extra value over $v_i^*$ or it contains all the elements of $v_i^*$ except one. We consider these two cases separately below.

(a) Let $v_i' = v_i^* \cup \{x_{iw}\}$ for some $x_{iw}$ not belonging to $v_i^*$.

$$g(v_1^*,\ldots,v_{N+M}^*) - g(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*)$$

$$= p_+P^+(v_1^*,\ldots,v_{N+M}^*) + p_-P^-(v_1^*,\ldots,v_{N+M}^*)$$

$$-p_+P^+(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*) - p_-P^-(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*)$$

$$= p_+\left[P^+(v_1^*,\ldots,v_{N+M}^*) - P^+(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*)\right]$$

$$+p_-\left[P^-(v_1^*,\ldots,v_{N+M}^*) - P^-(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*)\right] \tag{3.8}$$

Since $x_{iw} \notin v_i^*$ and $v_i' = v_i^* \cup \{x_{iw}\}$, we have $(v_i' - v_i^*) \cap v_i^* = \phi$ and $(v_i' - v_i^*) \cap (v_i^*)^c \neq \phi$.

Consequently, by monotone properties (M-1) and (M-2) of $P^+$ and $P^-$, the first term in (3.8) is zero and the second term is positive. Therefore,

$$g(v_1^*,\ldots,v_{N+M}^*) - g(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*) > 0$$

(b) Let $v_i' = v_i^* - \{x_{ic}\}$ for some $x_{ic} \in v_i^*$.

$$g(v_1^*,\ldots,v_{N+M}^*) - g(v_1^*,\ldots,v_{j-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*)$$

$$= p_+\left[P^+(v_1^*,\ldots,v_{N+M}^*) - P^+(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*)\right]$$

$$+p_-\left[P^-(v_1^*,\ldots,v_{N+M}^*) - P^-(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*)\right] \tag{3.9}$$

Since $x_{ic} \in v_i^*$ and $v_i' = v_i^* - \{x_{ic}\}$, we have $(v_i^* - v_i') \cap (v_i^*)^c = \phi$ and $(v_i^* - v_i') \cap v_i^* \neq \phi$.

Consequently, by (M-1) and (M-2), the first term in (3.9) is positive and the second term is zero. Therefore,

$$g(v_1^*,\ldots,v_{N+M}^*) - g(v_1^*,\ldots,v_{i-1}^*,v_i',v_{i+1}^*,\ldots,v_{N+M}^*) > 0$$

This proves Case 1.

*Case* 2

We have to show the existence of $\epsilon^* > 0$ such that $\forall 0 < \epsilon < \epsilon^*$,

$$g\left(v_1^*,\ldots,v_{N+M}^*\right) \geq g\left(v_1^*,\ldots,v_{i-1}^*,v_i''(\epsilon),v_{i+1}^*,\ldots,v_{N+M}^*\right), \quad \forall i = N+1,\ldots,N+M$$

where

$$v_i''(\epsilon) = \left[w_{i1}, w_{i2}\right]$$

with either $w_{i1} \in \left(w_{i1}^* - \epsilon, w_{i1}^* + \epsilon\right)$ and $w_{i2} = w_{i2}^*$ or $w_{i2} \in \left(w_{i2}^* - \epsilon, w_{i2}^* + \epsilon\right)$ and $w_{i1} = w_{i1}^*$. (It may be recalled that by our notation $v_i^* = \left[w_{i1}^*, w_{i2}^*\right]$.) We prove Case 2 only for the left end point and the proof will follow on similar lines for the right end point.

We consider the cases $w_{i1} \in \left(w_{i1}^* - \epsilon, w_{i1}^*\right)$ and $w_{i1} \in \left(w_{i1}^*, w_{i1}^* + \epsilon\right)$ separately below and prove Case 2.

(a) Let $w_{i1} \in \left(w_{i1}^* - \epsilon, w_{i1}^*\right)$.

$$\begin{aligned} & g\left(v_1^*,\ldots,v_{N+M}^*\right) - g\left(v_1^*,\ldots,v_{i-1}^*,v_i''(\epsilon),v_{i+1}^*,\ldots,v_{N+M}^*\right) \\ & \quad = p_+\left[P^+\left(v_1^*,\ldots,v_{N+M}^*\right) - P^+\left(v_1^*,\ldots,v_{i-1}^*,v_i''(\epsilon),v_{i+1}^*,\ldots,v_{N+M}^*\right)\right] \\ & \qquad + p_-\left[P^-\left(v_1^*,\ldots,v_{N+M}^*\right) - P^-\left(v_1^*,\ldots,v_{i-1}^*,v_i''(\epsilon),v_{i+1}^*,\ldots,v_{N+M}^*\right)\right] \end{aligned} \tag{3.10}$$

Since $w_{i1} < w_{i1}^*$ and $v_i^* \subset v_i''(\epsilon)$, we have $\left(v_i''(\epsilon) - v_i^*\right) \cap v_i^* = \phi$ and $\left(v_i''(\epsilon) - v_i^*\right) \cap \left(v_i^*\right)^c \neq \phi$. Consequently, by (M-1) and (M-2), the first term in (3.10) is zero and the second term is positive.

(b) Let $w_{i1} \in \left(w_{i1}^*, w_{i1}^* + \epsilon\right)$.

$$\begin{aligned} & g\left(v_1^*,\ldots,v_{N+M}^*\right) - g\left(v_1^*,\ldots,v_{i-1}^*,v_i''(\epsilon),v_{i+1}^*,\ldots,v_{N+M}^*\right) \\ & \quad = p_+\left[P^+\left(v_1^*,\ldots,v_{N+M}^*\right) - P^+\left(v_1^*,\ldots,v_{i-1}^*,v_i''(\epsilon),v_{i+1}^*,\ldots,v_{N+M}^*\right)\right] \\ & \qquad p_-\left[P^-\left(v_1^*,\ldots,v_{N+M}^*\right) - P^-\left(v_1^*,\ldots,v_{i-1}^*,v_i''(\epsilon),v_{i+1}^*,\ldots,v_{N+M}^*\right)\right] \end{aligned} \tag{3.11}$$

Since $w_{i1} > w_{i1}^*$ and $v_i^* \supset v_i''(\epsilon)$, we have $\left(v_i^* - v_i''(\epsilon)\right) \cap \left(v_i^*\right)^c = \phi$ and $\left(v_i^* - v_i''(\epsilon)\right) \cap v_i^* \neq \phi$. Consequently, by (M-1) and (M-2), the first term in (3.11) is positive and the second term is zero. Therefore,

$$g\left(v_1^*,\ldots,v_{N+M}^*\right) - g\left(v_1^*,\ldots,v_{i-1}^*,v_i'',v_{i+1}^*,\ldots,v_{N+M}^*\right) > 0$$

Since $\epsilon$ is arbitrary, Case (2) is proved.

Thus, $\left(v_1^*,\ldots,v_{N+M}^*\right)$ is an optimal point of $g(\cdot)$.

*Part 2*

We show here that $(v_1,\ldots, v_{N+M})$ is not an optimal point when $v_i$ are such that

- $v_i \neq v_i^*$, for at least one $i$
- $v_i \neq \phi$, $\forall i$

To show $(v_1,\ldots, v_{N+M})$ is not an optimal point we need to show that either

- $\exists v_i'$, for one $i$, $1 \leq i \leq N$, $v_i'$ differing from $v_i$ in exactly one element such that

$$g(v_1,\ldots, v_{i-1}, v_i', v_{i+1},\ldots, v_{N+M}) > g(v_1,\ldots, v_{N+M}),$$

(OR)

- For one $i$, $N+1 \leq i \leq M$, for all $\epsilon^* > 0$, $\exists 0 < \epsilon < \epsilon^*$ such that $\exists v_i'(\epsilon)$, $v_i'(\epsilon)$ identical to $v_i$ except for only one end point that is atmost $\epsilon$ distance from the corresponding end point of $v_i$ such that

$$g(v_1,\ldots,v_{i-1},v_i'(\epsilon),v_{i+1},\ldots,v_{N+M}) > g(v_1,\ldots,v_{N+M}),$$

In the following proof, whenever a linear attribute is dealt with, we consider only the left end point (i.e. $l = 1$) of any interval. The proof for the right end point will follow in similar steps.

To prove that the given concept $(v_1,\ldots, v_{N+M})$ is not an optimal point, we select a $v_i$ such that $v_i \neq v_i^*$ and $v_i \neq \phi$. Then, we only have the following four possibilities.

1. $\exists x_{iw}, x_{iw} \in v_i$ but $x_{iw} \notin v_i^*$, $I = 1,\ldots, N$.
2. $w_{i1} < w_{i1}^*$ or $w_{i2} > w_{i2}^*$ where $v_i = [w_{i1}, w_{i2}]$ and $v_i^* = [w_{i1}^*, w_{i2}^*]$ $i = N+1,\ldots, N+M$.
3. $\exists x_{ic}, x_{ic} \notin v_i$ but $x_{ic} \in v_i^*$, $i = 1,\ldots, N$.
4. $w_{i1} > w_{i1}^*$ or $w_{i2} < w_{i2}^*$ where $v_i = [w_{i1}, w_{i2}]$ and $v_i^* = [w_{i1}^*, w_{i2}^*]$, where $i = N+1,\ldots, N+M$.

Case 1

$\exists x_{iw}, x_{iw} \in v_i$ but $x_{iw} \notin v_i^*$

Let $v_i' = v_i - \{x_{iw}\}$

$$\begin{aligned} &g(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_{N+M}) - g(v_1,\ldots,v_{N+M}) \\ &\quad = p_+\left[P^+(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_{N+M}) - P^+(v_1,\ldots,v_{N+M})\right] \\ &\qquad + p_-\left[P^-(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_{N+M}) - P^-(v_1,\ldots,v_{N+M})\right] \\ &\qquad > 0, \end{aligned} \tag{3.12}$$

since, by (M-1) and (M-2), the first term is zero and the second term is positive.

*Case* 2

$w_{i1} < w_{i1}^*$

Pick $0 < \epsilon < (w_{i1}^* - w_{i1})/2$ and let $v_i''(\epsilon) = [w_{i1} + \epsilon, w_{i2}]$.

$$\begin{aligned} & g(v_1,\ldots,v_{i-1},v_i''(\epsilon),v_{i+1},\ldots,v_{N+M}) - g(v_1,\ldots,v_{N+M}) \\ & \quad = p_+\left[P^+(v_1,\ldots,v_{i-1},v_i''(\epsilon),v_{i+1},\ldots,v_{N+M}) - P^+(v_1,\ldots,v_{N+M})\right] \\ & \qquad + p_-\left[P^-(v_1,\ldots,v_{i-1},v_i''(\epsilon),v_{i+1},\ldots,v_{N+M}) - P^-(v_1,\ldots,v_{N+M})\right] \\ & \qquad > 0, \end{aligned} \tag{3.13}$$

since, by (M-1) and (M-2), the first term is zero and the second term is positive.

If $w_{i2} > w_{i2}^*$, we choose $v_i''(\epsilon) = [w_{i1}, w_{i2} - \epsilon]$, and the proof is similar.

*Case* 3

We can assume without loss of generality that

- no $v_i$ contains an $x_{iw}$ such that $x_{iw} \in v_i$ but $x_{iw} \notin v_i^*$, $1 \le i \le N$.
- $w_{i1}^* \le w_{i1}$ and $w_{i2}^* \ge w_{i2}$, $N + 1 \le i \le M$.

(Otherwise, we can use Case 1 or Case 2 to construct a $v_i$ or $v_i''(\epsilon)$ to satisfy (3.12) or (3.13) respectively.)

Hence, $v_i \subset v_i^*, \forall i$..

Now if for some $i$, $1 \le i \le N$, $v_i \ne v_i^*$, then $\exists x_{ic}$, $x_{ic} \notin v_i$ but $x_{ic} \in v_i^*$

Let $v_i' = v_i \bigcup \{x_{ic}\}$

$$\begin{aligned} & g(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_{N+M}) - g(v_1,\ldots,v_{N+M}) \\ & \quad = p_+\left[P^+(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_{N+M}) - P^+(v_1,\ldots,v_{N+M})\right] \\ & \qquad + p_-\left[P^-(v_1,\ldots,v_{i-1},v_i',v_{i+1},\ldots,v_{N+M}) - P^-(v_1,\ldots,v_{N+M})\right] \\ & \qquad > 0, \end{aligned}$$

since, by (M-1) and (M-2), the first term is positive and the second term is zero as $v_i \subset v_i^*, \forall i$.

Case 4. $w_{i1} > w_{i1}^*$

Pick $0 < \epsilon < (w_{i1} - w_{i1}^*)/2$ and let $v_i''(\epsilon) = [w_{i1} - \epsilon, w_{i2}]$.

$$g(v_1,\ldots,v_{i-1},v_i'(\epsilon),v_{i+1},\ldots,v_{N+M}) - g(v_1,\ldots,v_{N+M})$$
$$= \mu_1\left[P^+(v_1,\ldots,v_{i-1},v_i'(\epsilon),v_{i+1},\ldots,v_{N+M}) - P^+(v_1,\ldots,v_{N+M})\right]$$
$$+ \mu_2\left[P^-(v_1,\ldots,v_{i-1},v_i'(\epsilon),v_{i+1},\ldots,v_{N+M}) - P^-(v_1,\ldots,v_{N+M})\right]$$
$$< 0,$$

since, by (M-1) and (M-2), the first term is positive and the second term is zero as $v_i \subset v_i^*, \forall i$. This completes Part 2 of the proof and hence the theorem.

*Remark* 3.3. *By theorem* 2, *we know that the correct concept is an optimal point. Also, in all other optimal points of the payoff function at least one of the attributes has null set as correct set. Since, in a realistic concept learning problem where the 'right' concept is a simple conjunctive expression, the correct set cannot be a null set, it is easy to check whether the algorithm has converged to the correct concept. If at least one of the sets is null in the converged concept, then we can rerun the algorithm with a different starting point, i.e. with a different seed to the random number generator.*

In view of Lemma 1 and Theorems 1 and 2, Algorithm-1 essentially converges (by Remark 3.3) to the *correct* concept if the learning parameters are sufficiently small. As the distribution over the space of examples is arbitrary, Algorithm-1 correctly learns the class of simple disjunctive concepts under the PAC framework, upto concepts with null sets. Since we can prevent the convergence to concepts with null sets in view of Remark 3.3, the correct learning of Algorithm-1 can be ensured by making it automatically loop back till the converged concept has no null sets. Algorithm-1 indeed always converged to correct concept and never needed such looping back in all the simulation studies (See section 6) of our algorithm.

## 4. Learning Disjunctive Concepts

In this subsection, we see how we can extend Algorithm-1 to learn disjunctive concepts. Consider the learning problem where the target concept is representable as a $k$-term disjunctive concept (cf. Definition 2). Suppose we use the model of section 3.2 for this learning problem. Since each disjunct (i.e. each term) in the target concept is a simple conjunctive expression, we can expect the payoff function to have optimal points corresponding to each disjunct. Therefore, it seems we can think of an algorithm which would learn the target disjunctive concept by just calling Algorithm-1 $k$ times! However, this algorithm in general will not work because of two reasons. Firstly, there is no guarantee that each call to Algorithm-1 will return a different concept. So, even if we keep calling Algorithm-1 until we get $k$ different concepts, it is possible that this procedure runs indefinitely. Secondly, the payoff function may have optimal points not corresponding to any disjunct, in which case the algorithm might converge to a spurious disjunctive concept. For example, consider a domain consisting of one nominal and one linear attribute, say $A_1$ and $A_2$. Let $A_1$ take values in $\{A, B, C\}$ and $A_2$ take values from the interval [0.0, 5.0]. Let the target concept be

$$\{[A_1 \in \{A, B\}] \wedge [A_2 \in [3.0, 4.0]]\} \vee \{[A_1 \in \{B, C\}] \wedge [A_2 \in [1.0, 2.0]]\}.$$

Then, the conjunctive concepts $\{[A_1 \in \{A, B, C\}] \Lambda [A_2 \in [1.0, 4.0]]\}$ and $\{[A_1 \in \{A, C\}] \Lambda [A_2 \in [1.5, 3.5]]\}$ could be the optimal points of the payoff function for some probability distribution over the examples. We tackle these two problems as below.

The first problem can be solved as follows. After a disjunct of the target concept is learnt, we label all examples satisfied by this disjunct as negative. Consequently, this disjunct is no longer "visible" and so Algorithm-1 will output a new concept the next time. As for the second problem, at present we do not know whether it is solvable in general for all $k$-term disjunctive concepts. However, we propose a modification to our model of section 3.2 so that for a particular class of $k$-term disjunctive concepts, the payoff function possesses no spurious optimal points. This class of concepts we consider, to be called disjunctive concepts with a Marker attribute, is defined below.

Let the attributes chosen for the domain be $Y_i$ that take values from sets $V_i$, $i = 1,..., N$.

*Definition* 7. *A concept description given by*

$$[Y_1 \in v_1] \Lambda ... [Y_{m-1} \in v_{m-1}] \Lambda [Y_m = x_m] \Lambda [Y_{m+1} \in v_{m+1}] \Lambda ... \Lambda [Y_N \in v_N]$$

*where $v_i$ is a subset of $V_i$, $i = 1,..., N$, $i \neq m$, and $x_m \in V_m$ is said to be a* `simple conjunctive expression with a marker attribute`. *$Y_m$ is called the* marker *attribute.*

In the above definition, the subset $v_i$ for a linear attribute should be an interval.

*Definition* 8. *A concept description of the form*

$$C_1 \vee C_2 \vee ... \vee C_k$$

where each $C_i$ is a simple conjunctive expression with $Y_m$ as the marker attribute and all $C_i$–s have distinct values for the marker attribute, is said to be a `$k$-term disjunctive expression with a marker attribute`. *$Y_m$ is called the marker attribute for the concept.*

We assume that the attributes characterizing the domain contain at least one nominal attribute. If there are two or more nominal attributes, then we assume the teacher to specially identify one nominal attribute as the marker attribute. In case all the domain attributes are linear, then we properly discretize one of these attributes and use it as the marker attribute.

From Definition 8, we observe that in a disjunctive concept with a marker attribute, the marker attribute assumes distinct values in each disjunct of the concept. Now, by using the fact that these values serve as "markers" to the disjuncts in the target concept, we can get rid of spurious optimal points. Hence the name marker attribute.

Disjunctive concepts with a marker attribute constitute a special subclass of $k$-term disjunctive concepts. This class includes all simple conjunctive concepts and some non-trivial disjunctive concepts also[20]. However, we cannot represent disjunctive concepts like the 3-bit parity concept with a single marker attribute. We may possibly learn these concepts if there are two or more marker attributes. This problem is discussed by Rajaraman, *et al*[20].

Our model for learning disjunctive concepts with a marker attribute is as follows. In this model, we assume that the identity of marker attribute is provided by the teacher. We use the same model of section 3.2 but with a simple modification to handle marker attributes. In this model, we represent the marker attribute $Y_m$ by a FALA having $\left|V_m^{(d)}\right|$ actions. Denote this FALA by $X_m^{(d)}$. Each action choice by $X_m^{(d)}$ is interpreted as the selection of the corresponding value for $Y_m$. That is, if the FALA chooses the $l$-th action then $Y_m$ appears in the chosen concept as the equality predicate $[Y_m = x_{ml}]$, where $x_{ml}$ denotes the $l$-th value of $Y_m$. All non-marker attributes are represented as in section 3.2. The algorithm used by our model is same as Algorithm-1 except that the updating algorithm of $X_m^{(d)}$ representing $Y_m$ is modified as below.

Let $p_{mj}(k) = Pr[x_{mj}$ is chosen at $k$-th instant by the automaton $X_m^{(d)}]$, $j = 1,\ldots, \left|V_m^{(d)}\right|$, define the action probability distribution of $X_m^{(d)}$. Let $\alpha_m(k) = x_{ml}$ and the environmental response be $r'(k)$(cf. equation (3.5)). Then,

$$p_{ml}(k+1) = p_{ml}(k) + \lambda r'(k)(1 - p_{ml}(k))$$

$$p_{mj}(k+1) = p_{mj}(k) - \lambda r'(k) p_{mj}(k), \quad \forall j \neq l$$

We call this new learning algorithm used by the team as Algorithm-2. The payoff structure is as in (3.5). For this model, it can be proved similar to Theorem 2 that every disjunct in the target disjunctive concept with a marker attribute, is an optimal point and in all other optimal points, null set is the subset chosen for at least one non-marker attribute.

Now it is easy to prove[20] that we can learn disjunctive concepts by iteratively calling Algorithm-2. The choice of a value for the marker attribute can be utilized to correspond to exactly one disjunct in the target concept(or none at all) and so there are no spurious optimal points[20]. We present an outline of the learning algorithm for the disjunctive concepts with a marker attribute below.

Algorithm-3

- Repeat
  - LIST : = $\phi$
  - Repeat
    - * Get a random example. If it is positive and is satisfied by at least one concept in LIST, then label it negative. Run Algorithm-2 with this example.
  - Until convergence
  - Add the learnt concept to LIST
- Until $k$ concepts are accumulated in LIST
- Output LIST

## 5. Discussion

In this paper, we presented an algorithm for learning the class of simple conjunctive concepts involving both nominal and linear attributes. We proved that the algorithm correctly learns the class of simple conjunctive concepts(modulo concepts with null sets) under noise. The algorithm is incremental and hence needs no storage of examples. We also presented an extension to this algorithm which can learn a special class of disjunctive concepts.

We modelled noise by assuming an unknown distribution over the instance space and each example to be misclassified in an unbiased manner before being presented to the learning system. Because the examples are i.i.d in PAC formulation, we observed that this noise model is same as having an arbitrary distribution over the space of classified examples with the extra assumption that the examples undergo unbiased misclassifications. By the unbiasedness assumption, when the noise probability is less than 50%, it is reasonable to expect correct learning even under noise. However, as we assume passive learning setup (i.e. no queries), we cannot get to know the correct classification of an example with better accuracy by repeatedly asking the teacher as in Sakakibara[21], even if the noise probability is less than 50%. Our algorithm is able to handle noise by the stochastic nature of the search employed over the hypothesis space. Our noise model is closely related to that of Angluin *et al*[8]. who, as in our model, assume that the examples undergo unbiased misclassifications. However, strictly speaking, they require an upper bound on the noise probability be known to the learning algorithm. Our model assumes only that the noise probability is less than 0.5 and hence the algorithm has no additional overhead to estimate bounds on the noise probability.

A similar algorithm based on the model of team of automata has been used earlier for concept learning[13]. That model learns simple conjunctive concepts expressed through nominal attributes. It is also incremental and has provable generalization properties. However, the model cannot handle linear attributes and, as noted earlier, our algorithm can be thought of as a generalized version of Sastry *et al*[13].

Another important aspect of our model is that our algorithm is parallel. All automata choose actions independently based on their respective action probability distributions, get a common reinforcement from the environment and independently update the distributions. There is no explicit communication between the automata. Hence, employing one processor per automaton in a SIMD machine, we can expect almost a linear speedup.

Concept learning under noise has received much attention in AI and recently in Computational Learning Theory (COLT). The popular algorithms in AI are the Decision Tree based methods which include the Quinlan's ID3 algorithm[4,22] and its extensions[23,24] and the Classification and Regression Trees (CART) algorithm[25]. In COLT, the "$\in$ Version Space" Algorithm[7], $k$-CNF learning algorithm of Angluin *et al*[8]., Haussler's Empirical Risk Minimization algorithm[12] and $p$-concept learning algorithms of Kearns *et al*[9]. are some of the methods proposed for learning under noise.

Decision tree based methods constitute an important class of methods capable of handling noise. Quinlan's ID3 algorithm and the CART algorithm are two popular members of this class. Both these algorithms learn concepts in the form of decision trees by

recursively partitioning the training set. Quinlan's algorithm uses an information theoretic measure as a heuristic to construct compact trees. The CART algorithm also builds trees using heuristic measures but, more importantly, uses pruning techniques based on statistical considerations so that the learnt tree is not overly sensitive to the training examples. The algorithms are computationally efficient and found to be noise-tolerant[25,22,23]. Also, they can learn disjunctive concepts. However, they are non-incremental and hence need to store all the examples. Though incremental versions of the algorithms have been proposed[26], since they essentially try to minimize the number of "calls" to the basic algorithm, storing all the examples may still be necessary. The methods grow huge trees under noisy conditions and hence, as an additional overhead, it may be necessary to prune the learnt tree. Also, we are not aware of any theoretical result on the PAC learnability of these algorithms under noise.

The $\in$ version space algorithm[7] is a modified form of Mitchell's candidate elimination algorithm[27] proposed by Haussler. Imposing a partial order over the hypothesis space, as in Mitchell[27], the algorithm learns by updating the so-called version space of the examples seen so far. But unlike Mitchell's algorithm, it uses PAC identification as termination criterion. The algorithm is incremental and is proved to efficiently PAC learn concepts which use only boolean attributes. However, the algorithm may be space inefficient for learning rich concept classes and, in particular, it cannot handle linear attributes.

Angluin *et al.* are one of the first to analyze the problem of PAC learning under noise. They propose an algorithm[8] for learning $k$-CNF concepts and prove that it efficiently PAC learns under noise. As in our case, the algorithm can tackle upto 50% of classification noise. However, the algorithm is efficient only in finite domains and cannot handle linear attributes. Also, the algorithm is nonincremental.

Haussler[12] proposed a very general and powerful framework for PAC learning, which can take care of noise, using decision theoretic ideas. His approach is based on viewing learning process as one of minimizing empirical risk. We have used this framework in section 2.1 to define the goal of our learning system. Under this framework, he proves bounds on the sample size needed for PAC learning very general concept classes including the class of artificial neural networks. However, his algorithm is not computationally efficient.

The work of Kearns *et al.* attempts to apply some of Haussler's general principles to a specific setting so that efficient algorithms can be developed. They propose[9] several learning algorithms for PAC learning under noise and here we discuss only those relevant in our context. We also assume that all $p$-concepts learnt are converted into decision rules by suitably thresholding the $p$-concepts. First, an algorithm for learning the class of non-decreasing functions is presented. Under the assumption that the instance space is totally ordered, the algorithm is proved to efficiently PAC learn under noise. However, this assumption precludes effective handling of nominal attributes and so we do not discuss this any further. Learning a class of probabilistic decision lists is then considered. They propose an efficient algorithm for PAC learning under noise. But, the algorithm assumes the domain to be finite and hence cannot learn linear attributes. Finally, an algorithm for learning a linear function space of finite pseudodimension is presented. Each concept in this class is expressible as a linear combination of finite number of fixed functions. This

specific representation is chosen so that the method of minimizing empirical loss is computationally efficient. For the same reason, the algorithm cannot be extended to learn richer concept classes, e.g. concepts expressible as a linear combination of functions from an infinite class. Moreover, all these algorithms are nonincremental.

To summarize, the algorithms in AI are empirically efficient and have been employed to solve realistic problems. Some of these algorithms are found to handle noise in empirical studies. However, their noise-tolerance in a general concept learning problem is unknown because they lack PAC learnability results under noise. Also, barring a few, the algorithms are not truly incremental. On the COLT side, the learning algorithms have provable convergence properties and are shown to be theoretically efficient. Their noise-tolerance has also been precisely characterized. However, almost all these algorithms assume unrealistic domains, e.g. domain characterized by boolean-valued attributes. Hence, the COLT algorithms turn out to be mainly of theoretical interest. The main motivation behind our work is to fill this gap so that we can come up with realistic, practically efficient, robust and incremental algorithms which PAC learn even in the presence of noise. The next section justifies this where we present empirical results of our algorithm.

## 6. Simulation Studies

In this section, we present empirical studies of Algorithm-1 and Algorithm-3 on few synthetic and real-world domains.

### 6.1. *Synthetic Problems*

A synthetic domain is defined by specifying the attributes (nominal and/or linear) and their ranges. Then, an arbitrary concept description in the chosen representation (a simple conjunctive expression or a $k$-term disjunctive expression) involving the specified attributes is selected. A fixed number of pre-classified examples (called the the training set) is generated randomly according to a predefined probability distribution. To evaluate the performance of the algorithm, we generate another set of examples (called the test set) with respect to the same probability distribution. The sets are generated in such a way that the number of positive and negative examples are equal in each set.

We use Algorithm-1 for learning simple conjunctive concepts and Algorithm-3 for learning disjunctive concepts. The algorithms are simulated by selecting one example at a time from the training set and if needed, using the training set repeatedly for drawing examples till convergence. The performance is studied by varying the classification noise from 0% to 40%. For comparison, we also implemented a decision tree based algorithm[24] that can learn disjunctive concepts and handle noise. We refer to this algorithm as Algorithm-4.

The results of the two algorithms on two synthetic problems are presented below. The results of Algorithms 1 and 3 are given in Tables I, III and V, VII respectively. In these tables, $\lambda$ refers to the value of the learning coefficient used. The variance parameter $\sigma_L$ is set to a value of 0.1 in all the simulations. The column 'Average Iterations' refers to the average, over 20 runs of the algorithm, of the number of iterations taken to converge. The acronym WC refers to the number of runs (out of 20) in which the algorithms did not converge to the correct concept. The column Error rate refers to the percentage of examples

**Table I**
**Performance of algorithm-1**

| *Noise %* | *λ* | *Average Iterations* | *WC* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 0.005 | 9000 | 0 | 10 | 4.0 |
| 5 | 0.005 | 11000 | 0 | 13 | 5.0 |
| 10 | 0.005 | 14000 | 1 | 17 | 5.0 |
| 20 | 0.005 | 15500 | 1 | 20 | 6.0 |
| 30 | 0.005 | 18000 | 2 | 23 | 8.0 |
| 40 | 0.004 | 22000 | 3 | 28 | 11.0 |

**Table II**
**Performance of algorithm-4**

| *Noise %* | *Nodes* | *Leaves* | *Average Depth* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 10 | 11 | 4.1 | 0.5 | 4.0 |
| 5 | 12 | 16 | 4.8 | 1.0 | 6.0 |
| 10 | 16 | 17 | 5.2 | 1.0 | 9.0 |
| 20 | 32 | 33 | 7.3 | 2.0 | 19.0 |
| 30 | 41 | 42 | 6.9 | 2.0 | 23.0 |
| 40 | 43 | 44 | 9.3 | 3.0 | 29.0 |

misclassified by the algorithms on the test set. The time taken by the algorithm is given under the column CPU Time. The performance of Algorithm-4 is given in Tables II, IV, VI and VIII. In these tables, the columns Nodes,Leaves and Average Depth refer respectively to the number of nodes, number of leaves and the average depth of the learnt decision tree. These quantities give a measure of the size of the learnt tree.

The simulations were performed on an i860 based system.

### 6.1.1. *Problem* 1

Let the domain be characterized by 2 nominal and 2 linear attributes. Denote them by $A_1$, $A_2$, $A_3$ and $A_4$ respectively. The nominal attributes $A_1$ and $A_2$ take values in $\{A, B, C, D\}$ and the linear attributes $A_3$ and $A_4$ from [0.0, 5.0]. Let the target conjunctive concept be

$$[\{A_1 \in \{A, C\}\} \Lambda \{A_2 \in \{B, D\}\} \Lambda \{A_3 \in [2.0, 4.0]\} \Lambda \{A_4 \in [2.0, 4.0]\}]$$

*Case* 1: Training set size = 100; Test set size = 100;

Performance of Algorithm-1: See Table I.
Performance of Algorithm-4: See Table II.

*Case* 2: Training set size = 500; Test set size = 100;

Performance of Algorithm-1: See Table III.
Performance of Algorithm-4: See Table IV.

### 6.1.2. *Problem* 2

Let the domain consist of 2 nominal ($A_1$ and $A_2$) and 2 linear ($A_3$ and $A_4$) attributes as in Problem 1. Let the target concept be

**Table III**
**Performance of algorithm-1**

| *Noise %* | *λ* | *Average Iterations* | *WC* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 0.005 | 12500 | 0 | 14 | 2.0 |
| 5 | 0.005 | 15000 | 0 | 17 | 2.0 |
| 10 | 0.005 | 17000 | 0 | 20 | 3.0 |
| 20 | 0.005 | 20000 | 1 | 24 | 3.0 |
| 30 | 0.005 | 25000 | 2 | 29 | 4.0 |
| 40 | 0.005 | 31000 | 2 | 35 | 6.0 |

**Table IV**
**Performance of algorithm-4**

| *Noise %* | *Nodes* | *Leaves* | *AveragDepth* | *CPU Tine Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 15 | 17 | 4.6 | 2.0 | 3.0 |
| 5 | 17 | 18 | 5.4 | 5.0 | 4.0 |
| 10 | 24 | 25 | 6.8 | 18.0 | 8.0 |
| 20 | 39 | 30 | 7.3 | 28.0 | 14.0 |
| 30 | 55 | 56 | 8.2 | 44.0 | 22.0 |
| 40 | 61 | 62 | 8.9 | 58.0 | 28.0 |

**Table V**
**Performance of algorithm-3**

| *Noise %* | $\lambda$ | *Average Iterations* | *WC* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 0.005 | 15000 | 0 | 38 | 3.0 |
| 5 | 0.005 | 21000 | 0 | 57 | 5.0 |
| 10 | 0.005 | 25000 | 1 | 68 | 5.0 |
| 20 | 0.005 | 32000 | 2 | 95 | 12.0 |
| 30 | 0.005 | 40500 | 2 | 111 | 15.0 |
| 40 | 0.004 | 46000 | 3 | 133 | 17.0 |

**Table VI**
**Performance of algorithm-4**

| *Noise %* | *Nodes* | *Leaves* | *Average Depth* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 13 | 16 | 5.5 | 1.0 | 8.0 |
| 5 | 14 | 14 | 5.2 | 1.0 | 9.0 |
| 10 | 28 | 23 | 7.3 | 1.5 | 17.0 |
| 20 | 30 | 32 | 7.4 | 2.0 | 25.0 |
| 30 | 33 | 35 | 9.1 | 2.5 | 28.0 |
| 40 | 34 | 37 | 9.8 | 2.5 | 32.0 |

$$[\{A_1 \in \{A\}\} \wedge \{A_2 \in \{B, C\}\} \wedge \{A_3 \in [2.0, 4.0]\} \wedge \{A_4 \in [2.0, 4.0]\}]$$
$$\vee [\{A_1 \in \{B\}\} \wedge \{A_2 \in \{C, D\}\} \wedge \{A_3 \in [1.0, 3.0]\} \wedge \{A_4 \in [1.0, 3.0]\}]$$

and let $A_1$ be the marker attribute.

*Case* 1: Training set size = 100; Test set size = 100;

Performance of Algorithm-3: See Table V.
Performance of Algorithm-4: See Table VI.

*Case* 2: Training set size = 500; Test set size = 100;

Performance of Algorithm-3 See Table VII.
Performance of Algorithm-4: See Table VIII.

### 6.2. *Real-world Problems*

We consider two problems in the popular Iris Plants domain. This domain contains three types of plants namely Iris-setosa, Iris-versicolor and Iris-viginica. Each plant is characterized by four linear attributes *viz.* petal width, petal length, sepal width and sepal length. It is known[2] that the class Iris-setosa is linearly separable from the other two and the other two are not linearly separable. We consider the following nonlinearly separable classification problems

1. To classify whether the given plant is Iris-viginica or not
2. To classify whether the given plant is one of Iris-setosa and Iris-viginica or otherwise.

Since the domain consists of 150 examples only, we divide the data into two sets and use the first one as training set and the second as test test. The division is done carefully to

**Table VII**
**Performance of algorithm-3**

| *Noise %* | $\lambda$ | *Average Iterations* | *WC* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 0.005 | 14500 | 0 | 37 | 1.0 |
| 5 | 0.005 | 22500 | 0 | 61 | 1.0 |
| 10 | 0.005 | 27000 | 0 | 74 | 4.0 |
| 20 | 0.005 | 36000 | 1 | 105 | 6.0 |
| 30 | 0.005 | 43000 | 2 | 124 | 9.0 |
| 40 | 0.005 | 49000 | 2 | 142 | 11.0 |

**Table VIII**
**Performance of algorithm-4**

| *Noise %* | *Nodes* | *Leaves* | *Average Depth* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 15 | 16 | 5.4 | 2.5 | 4.0 |
| 5 | 17 | 19 | 6.1 | 15.0 | 6.0 |
| 10 | 34 | 41 | 7.7 | 28.0 | 8.0 |
| 20 | 52 | 55 | 8.2 | 38.0 | 18.0 |
| 30 | 96 | 87 | 10.2 | 54.0 | 25.0 |
| 40 | 128 | 111 | 13.4 | 66.0 | 28.0 |

**Table IX**
**Performance of algorithm-1**

| *Noise %* | $\lambda$ | *Average Iterations* | *WC* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 0.005 | 4500 | 0 | 5 | 2.6 |
| 5 | 0.005 | 6100 | 0 | 7 | 4.0 |
| 10 | 0.005 | 7900 | 0 | 8 | 4.0 |
| 20 | 0.005 | 10200 | 1 | 11 | 4.0 |
| 30 | 0.005 | 14100 | 2 | 16 | 5.3 |
| 0 | 0.004 | 25300 | 2 | 28 | 5.3 |

keep the distribution of classes same in both sets. We also train the algorithms with a set of examples (called duplicated training set) containing five copies each example in the original training set. Hence, the duplicated training set has size five times that of the original set. We study the performance of the algorithms by adding noise to the training set externally.

The results of simulation of Algorithm 1 and 3 are given in tables IX and XI respectively. Tables X and XII contain the results for Algorithm-4. In the latter, the column CPU Time-I refers to the time taken by the algorithm on the original training set and CPU Time-II the time taken on the duplicated training set. All other columns remain the same as the duplicated set contains no new examples. However, there is no change in the execution time for algorithms 1 and 3.

6.2.1. *Problem* 1

Performance of Algorithm-1: See Table IX.
Performance of Algorithm-4: See Table X .

6.2.2. *Problem* 2

The classification problem here is more difficult than Problem 1 because a plant in this problem may be positive if it is either Iris-setosa or Iris-viginica. We use a 2-term disjunctive concept to represent concepts in algorithm 3. Since all attributes are linear, we need to discretize one of the attributes to use as a marker attribute. It is known that the attribute sepal width and sepal length are relevant to the problem[2]. We choose sepal length for the discretization. A cutpoint of 1.0 was chosen by trial and error to discretize this attribute into a two valued nominal attribute. Thus, the new learning problem has 1 nomi-

**Table X**
**Performance of algorithm-4**

| *Noise %* | *Nodes* | *Leaves* | *Average Depth* | *CPU Tine Secs.* | *Error rate %* | *CPU Time-II Secs.* |
|---|---|---|---|---|---|---|
| 0 | 4 | 5 | 2.4 | 0.5 | 5.3 | 2.0 |
| 5 | 8 | 9 | 3.7 | 0.5 | 8.0 | 4.0 |
| 10 | 17 | 18 | 5.9 | 1.0 | 8.0 | 9.0 |
| 20 | 23 | 24 | 6.7 | 2.0 | 24.0 | 10.0 |
| 30 | 31 | 32 | 7.3 | 2.5 | 32.0 | 10.0 |
| 40 | 46 | 47 | 7.4 | 2.5 | 48.0 | 12.0 |

**Table XI**
**Performance of algorithm-3**

| *Noise %* | $\lambda$ | *Average Iterations* | *WC* | *CPU Time Secs.* | *Error rate %* |
|---|---|---|---|---|---|
| 0 | 0.004 | 5600 | 0 | 12 | 2.6 |
| 5 | 0.004 | 9300 | 0 | 26 | 4.0 |
| 10 | 0.004 | 10700 | 1 | 39 | 4.0 |
| 20 | 0.004 | 18100 | 1 | 55 | 5.3 |
| 30 | 0.003 | 22000 | 2 | 71 | 5.3 |
| 40 | 0.003 | 27200 | 3 | 86 | 8.0 |

nal and 3 linear attributes. The sizes of training as well as test test are same as in Problem 1.

Performance of Algorithm-3: See Table XI.
Performance of Algorithm-4: See Table XII.

### 6.3. *Discussion*

It can be observed that, though Algorithm-4 performs reasonably well under no noise conditions, the performance degrades as the noise percentage increases. Under the same conditions, Algorithms 1 and 3 exhibit better accuracy and, as the number of training examples is increased, the algorithms generalize well resulting in a further reduced error rate (cf. Tables I, III and V, VII). It appears that Algorithm-4 is computationally very efficient compared to Algorithms 1 and 3, especially under noise. However, this is true only when the number of training examples is small. As seen from tables VI and VIII, the execution time of Algorithm-4 drastically increases when the size of the training set in increased. Actually, this behaviour happens for a training set containing only duplicated copies of the training examples used for the previous problem (cf. Tables X, XII). This is the main disadvantage of nonincremental algorithms. On the other hand, Algorithms 1 and 3, being incremental, are not sensitive to such redundancy in the training set. Even if the size of the training set is increased to contain new examples, these incremental algorithms can effectively handle the situation and show no appreciable change in the execution time (cf. Tables I, III and V, VII).

## 7. Conclusion

We considered the problem of learning conjunctive and disjunctive concepts using a set of positive and negative examples of the concept. We formulated this problem in the Proba-

**Table XII**
**Performance of algorithm-4**

| *Noise %* | *Nodes* | *Leaves* | *Average Depth* | *CPU Tine Secs.* | *Error rate %* | *CPU Time-II Secs.* |
|---|---|---|---|---|---|---|
| 0 | 7 | 8 | 3.6 | 1.0 | 8.0 | 2.5 |
| 5 | 10 | 11 | 3.9 | 1.0 | 13.3 | 3.0 |
| 10 | 12 | 13 | 4.3 | 1.5 | 17.3 | 9.0 |
| 20 | 26 | 27 | 6.2 | 2.0 | 22.6 | 14.0 |
| 30 | 39 | 40 | 7.1 | 2.5 | 34.6 | 19.0 |
| 40 | 38 | 42 | 7.2 | 3.0 | 41.3 | 21.0 |

bly Approximately Correct Learning framework and presented an algorithm for learning simple conjunctive concepts. We proved that the algorithm correctly learns the class of conjunctive concepts in the presence of upto 50% of classification noise. We proposed an extension to this algorithm for learning a class of disjunctive concepts. From the simulation studies, it was observed that the algorithms are reasonably efficient for learning logic expressions. The slowness of our algorithms observed under some cases may be due to the fact that we simulated these parallel algorithms on a sequential machine. The speed could be improved through a parallel implementation of these algorithms.

## References


1. RADA, R. AND FORSYTH, R. — *Machine Learning: Applications to expert systems and information retrieval*, 1986, Ellis Horwood, Chichester, UK.
2. DUDA, R. O. AND HART, P. E. — *Pattern classification and scene analysis*, 1973, Wiley, New York.
3. MICHALSKI, R. S. — A theory and methodology of inductive learning, 1983, *Artif. Intell.*, 1983, **20**, 111–161.
4. QUINLAN, J. R. — Learning efficient classification procedures and their application to chess end games, In *Machine learning: An artificial intelligence approach*, (Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. eds), 1983, Tiogo, Palo Alto, California.
5. MICHALSKI, R. S., CARBONELL, J. G. AND MITCHELL, T. M. (EDS) — *Machine learning: An artificial intelligence approach*, 1983, Vol. 1, Tiogo, Palo Alto, California.
6. MICHALSKI, R. S., CARBONELL, J. G. AND MITCHELL, T. M. (EDS) — *Machine learning: An artificial intelligence approach*, 1986, Vol. 2, Morgan Kaufmann, Los Alto, California.
7. HAUSSLER, D. — Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artif. Intell.*, 1988, **36**, 177–221.
8. ANGLUIN, D. AND LAIRD, P. — Learning from noisy examples. *Machine learning*, 1988, **2**, 343–370.
9. KEARNS, M. J. AND SHAPIRO, R. E. — Efficient distribution-free learning of probabilistic concepts. In *Computational learning theory and natural learning systems* (S. J. Hanson, G. A. Drastal and R. L. Rivest eds), 1994, MIT Press, Cambridge.
10. VALIANT, L. G. — A theory of the learnable. Communications *of the ACM*, 1984, 1134–1142.
11. BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D. AND WARMUTH, M. K. — Learnability and the Vapnik Chervonenkis dimension. *J. Assoc. Comp. Machinery*, 1989, **36**, 929–965.
12. HAUSSLER, D. — Decision theoretic generalization of the PAC model for neural net and learning applications. *Information and Computation*, 1992, **100**, 78–150.
13. SASTRY, P. S., RAJARAMAN, K. AND RANJAN, S. R. — Learning optimal conjunctive concepts through a team of stochastic automata. *IEEE Trans. on Systems, man and cybernetics*, 1993, **23**, 1175–1184.
14. THATHACHAR, M. A. L. AND SASTRY, P. S. — Learning optimal discriminant functions through a cooperative game of automata, *IEEE Trans. on systems, man and cybernetics*, 1987, **17**(1), 73–85.

15. RANJAN, S. R. *Stochastic automata models for concept learning*, 1989, Bangalore, India, M. Sc. Thesis, Dept. of Electrical Engineering, Indian Institute of Science.

16. NARENDRA, K. S. AND THATHACHAR, M. A. L. *Learning automata: An introduction*, 1989, Prentice Hall, Englewood Cliffs.

17. SANTHARAM, G. *Distributed Learning with connectionist models for optimisation and control*, 1994, Ph. D. thesis, Dept. of Electrical Engineering, Indian Institute of Science, Bangalore, India.

18. SASTRY, P. S., PHANSALKAR, V. V. AND THATHACHAR, M. A. L. Decentralised learning of Nash equilibria in multi-person stochastic games incomplete information, *IEEE Trans. on systems, man and cybernetics*, 1994, 24, 769–777.

19. RAJARAMAN, K. AND SASTRY, P. S. Stochastic optimisation over continuous and discrete variables with applications to concept learning under noise, (under review)

20. RAJARAMAN, K. Robust distribution free learning of logic expressions, 1996, Ph. D. Thesis, Dept. of Electrical Engineering, Indian Institute of Science, Bangalore.

21. SAKAKIBARA, Y. On learning from queries and counterexamples in the presence of noise, *Information processing letters*, 1991, 37, 279–284.

22. QUINLAN, J. R. Effect of noise on concept learning. In *Machine learning: An artificial intelligence approach* (R. S. Michalski, J. G. Carbonell and T. M. Mitchell, eds), 1986, Morgan Kaufmann, Los Altos, California.

23. QUINLAN, J. R. Probabilistic decision trees. In *Machine learning: An artificial intelligence approach* (Yves Kodratoff and R. S. Michalski, eds), 1990, Morgan Kaufmann, San Mateo.

24. FAYYAD, U. M. AND IRANI, K. B. On the handling of continuous valued attributes in decision tree generation, *Machine learning*, 1992, 8, 87–102.

25. BREIMAN, L., FRIEDMAN, J. H. OLSHEN, R. A. AND STONE, C. J. *Classification and regression trees*, 1984, Belmont, Wadsworth.

26. CRAWFORD, S. L. Extensions to the CART algorithm, *Intl J. Man-machine studies*, 1989, 31, 197–217.

27. MITCHELL, T. M. Generalization as search, *Artificial intelligence*, 1982, 18(2), 203–226.