Stochastic Optimization Over Continuous and Discrete Variables with Applications to Concept Learning Under Noise

K. Rajaraman and P. S. Sastry

***Abstract*—We consider optimization problems where the objective function is defined over some continuous and some discrete variables, and only noise corrupted values of the objective function are observable. Such optimization problems occur naturally in PAC learning with noisy samples. We propose a stochastic learning algorithm based on the model of a *hybrid* team of learning automata involved in a stochastic game with incomplete information to solve this optimization problem and establish its convergence properties. We then illustrate an application of this automata model in learning a class of conjunctive logic expressions over both nominal and linear attributes under noise.**

***Index Terms*—Concept learning, learning automata, ODE analysis of learning algorithms, optimization, PAC learning, risk minimization.**

I. Introduction

Many learning problems involve optimization of an unknown functional. For instance, in pattern classification [1], the interest is in finding a discriminant surface (of some fixed form) that minimizes the probability of misclassification though the statistics of the pattern classes may be unknown. Similarly, many learning problems in adaptive control, signal processing, and concept learning can be looked upon as the optimization of a suitably defined functional. In this paper we present an algorithm, based on learning automata models [2], that is suitable for tackling such stochastic optimization problems. An interesting feature of the algorithm is that the functional to be optimized may be defined over some continuous and some discrete variables and thus it is attractive for, e.g., concept learning problems as we illustrate toward the end of the paper.

Tsypkin [3] is among the first to formalize the unifying view of learning problems as the optimization of a performance index

$$J(\mathbf{w}) = \int_X R(\mathbf{x}, \mathbf{w})\, dP \tag{1.1}$$

where $R(\mathbf{x}, \mathbf{w})$ is a functional of the parameter vector $\mathbf{w}$ and observations $\mathbf{x}$, X is the space of all vectors $\mathbf{x}$, and P is a probability measure on X. The distinguishing feature of learning problems is that P is unknown. As an example, the 2-class pattern recognition problem can be formulated by choosing

$$R(\mathbf{w}, \mathbf{x}) = I\{f(\mathbf{w}, \mathbf{x}) > 0\}$$

where $f(., \mathbf{x})$ is the class of discriminant functions parameterized by $\mathbf{w}$ and $\mathbf{x}$ is the feature vector.[1]

The recent work in statistical learning theory and probably approximately correct (PAC) learning (see, e.g., [4] and [5]) once again makes explicit this connection between learning and optimization. To concretize a few details and to place the results here in perspective, we briefly describe an extension of the Tsypkin formulation along these lines following [5]. Consider a learning system interacting with a teacher. The aim is to learn an unknown concept (or function) using examples provided by the teacher. Each example consists of an instance $x \in X$ and an outcome $y \in Y$, where X and Y are called *instance* and *outcome* spaces, respectively, and may be arbitrary sets. The examples are generated in an independent and identically distributed (i.i.d.) manner according to an unknown probability distribution P_{XY}, defined on $X \times Y$. Using these examples, the learning system outputs a hypothesis from a *hypothesis* space H based on a learning algorithm $\mathcal{A}$. Any hypothesis $h \in H$ is a mapping from the instance space X to a *decision* space A, which again may be an arbitrary set. The *error* of any hypothesis is measured through a *loss* function, l: $Y \times A \to \Re$. $l(y, h(x))$ is the loss suffered by the hypothesis h on an example (x, y). The loss function is assumed to be known to the learner. The objective for the learner is to choose a hypothesis to minimize the expected loss. Formally, define

$$L(h) = E[l(y, h(x))] \tag{1.2}$$

$$h^* = \arg\min_h L(h) \tag{1.3}$$

where the expectation in (1.2) is w.r.t. P_{XY}, and h^* is the "correct" concept having the minimal expected loss (or risk). $L(\cdot)$ is called the risk function.

Definition 1.1: Let X, Y, P_{XY} 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 processed. Then the algorithm *probably approximately correctly* (PAC) *learns* H if $L(h_n)$ converges to $L(h^*)$ (in some

[1] $I\{A\}$ is the indicator function of the event A.

Manuscript received September 28, 1998; revised July 26, 1999. This paper was recommended by Associate Editor P. Borne.

K. Rajaraman is with the Kent Ridge Digital Labs, Singapore 119 613 (e-mail: kanagasa@krdl.org.sg).

P. S. Sastry is with the Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India.

Publisher Item Identifier S 1083-4427(99)08396-4.

suitable stochastic sense) as n goes to infinity, irrespective of the distribution P_{XY}.

If the functions in H can be parameterized by a vector of reals, then $J(\cdot)$ defined by (1.1) is essentially the same as $L(\cdot)$ defined by (1.2) and thus the above definition generalizes the classical Tsypkin formulation. This is also a generalization of the original PAC notion of concept learning and it also takes care of the case of noisy samples since we assumed that P_{XY} is an arbitrary distribution on $X \times Y$. (See [5] for a discussion.)

The learning problem now is to identify (or approximate) h^* defined by (1.3) given only a sample $\{(x_i, y_i), 1 \leq i \leq m\}$ of i.i.d. examples drawn according to P_{XY}. It may be noted that given an $h \in H$, $L(h)$ is not available because P_{XY} is unknown though for any random example (x, y), we can observe $l(y, h(x))$. A common strategy used in statistical learning theory for approximating h^* is the so called empirical risk minimization [4]. Define a functional $\hat{L}_m$ on H, called empirical risk, and $\hat{h}_m^* \in H$ by

$$\begin{aligned}\hat{L}_m(h) &= \hat{E}_m[l(y, h(x))] \\ &= \frac{1}{m}\sum_{i=1}^{m} l(y_i, h(x_i)) \end{aligned} \tag{1.4}$$

$$\hat{h}_m^* = \arg\min_h \hat{L}_m(h) \tag{1.5}$$

where $\hat{E}_m$ denotes expectation with respect to the empirical distribution defined by the sample (of m examples). Now suppose that the sequence of functions $\hat{L}_m(\cdot)$ converges to $L(\cdot)$ uniformly over H. Then for sufficiently large m, $\hat{h}_m^*$ will be a good approximator to h^*. (See [4] for an excellent discussion of this issue.) Thus we can think of two subparts to a learning problem: statistical and optimization [4], [5]. Ensuring that the needed uniform convergence holds for the chosen hypothesis space and obtaining some good bounds on the number of examples needed for approximating h^* to a desired degree constitutes the statistical part. It essentially deals with the question of whether certain conclusions drawn from a given finite sample generalize well to the whole population. Finding (or approximating) the actual minimizer of the empirical risk (or the actual risk) constitutes the optimization subpart which is the focus of this paper. Most of the computational learning theory literature concentrates on the statistical part and considerable attention has been paid to investigating, for example, the sample complexities of many useful hypotheses spaces [5]–[7]. For solving the full learning problem, we also need to guarantee that algorithm $\mathcal{A}$ can search the space H to find the optimizer of empirical risk. For simple concept learning problems (mostly over nominal attributes) with noise-free samples, many learning algorithms are available (e.g., [6] and [7]). But for more complex hypothesis spaces or for examples drawn according to an arbitrary P_{XY} (and thus allowing for any general noise) this optimization problem is rather difficult.

Given the set of m examples, $\hat{L}_m(h)$ can be calculated for any $h \in H$ and hence we can, in principle, employ a standard optimization technique to find $\hat{h}_m^*$ if H has a nice structure (for example, if H is isomorphic to some finite dimensional Euclidean space) even though in practice it can be a difficult nonlinear optimization problem. When each $h \in H$ is represented by a real vector, we can also use regression function learning algorithms such as stochastic approximations [8] to asymptotically approximate h^* [by observing $l(y, h(x))$ on the i.i.d. samples] if the functions h and l are well behaved. When there are only finitely many samples which are repeatedly used by uniformly sampling them, the stochastic approximation algorithms also amount to minimizing empirical risk [4].

The motivation behind our approach is that in many concept learning problems, elements of H are naturally parameterized by a vector of both continuous and discrete variables and thus H may have no simple algebraic structure on it. Further, any optimization technique that relies on some sort of estimated gradient may encounter numerical problems if the loss function is discontinuous.

In this paper we propose a regression function learning algorithm based on learning automata (LA) models [2] which can be used for minimizing the empirical risk without needing a nice algebraic structure on H and which does not explicitly estimate any gradients. The essence of the automata approach is the following. We set up a specific system of learning automata such that state of this system at any time represents a probability distribution over H. At each instant we choose an $h \in H$ at random based on the current probability distribution. Then the value of $l(y, h(x))$, where (x, y) is the next random example, is used as *reinforcement signal* to update the probability distribution using a learning algorithm. In this paper we propose a learning algorithm and show that it converges to a distribution that assigns an arbitrarily high probability to a hypothesis in H that is arbitrarily close to a (local) minimizer of the empirical risk (or the expected risk if there is an infinite sequence of examples i.i.d. according to P_{XY}). It is this strategy of searching in the space of probability distributions over H (rather than in H directly) that obviates the necessity of any algebraic structure on H making the LA approach attractive. In the remaining part of this section we give a very brief overview of learning automata mainly to introduce our notation.

Learning automata are adaptive decision making units that learn to choose the optimal action from a set of actions by interacting with a random environment [2]. The learning automaton maintains, at each instant k, a probability distribution over the action set, say $\mathbf{p}(k)$. The action at instant k, $a(k)$, is chosen at random based on this distribution. For each action choice, the environment provides a stochastic response (which is a scalar) called the *reinforcement*. This is used to update $\mathbf{p}(k)$ into $\mathbf{p}(k+1)$ by employing a learning algorithm.

Two types of learning automata, namely, finite action set learning automata (FALA) and continuous action set learning automata (CALA) are distinguished based on whether the action set is finite or it is the real line [9], [10]. For a FALA with, say, r actions, the action probability distribution is represented by an r-dimensional probability vector and this is updated by the learning algorithm. For CALA (whose action set is $\Re$), we represent the action probability distribution by a normal distribution. At each instant the learning algorithm updates the mean and variance of this distribution.

For an r-action FALA, let $\mathbf{p}(k) = [p_1(k) \cdots p_r(k)]$ be the action probability distribution at instant k. Let d_i, called reward probability of ith action, denote the expected value of reinforcement when ith action is chosen. If $d_l = \max_i\{d_i\}$ then we desire that the learning algorithm make $\mathbf{p}(k)$ converge to a probability vector, $\mathbf{p}$, with p_l arbitrarily close to unity. For a CALA (whose action set is $\Re$), let r_x denote the stochastic reinforcement when the action selected is $x \in \Re$. Then we define the so called reward function by $f(x) = E[r_x]$. The CALA has to maximize $f(\cdot)$ by observing r_x. Let $(\mu(k), \sigma(k))$ be the mean and standard deviation of the normal distribution which represents the action probability distribution of the CALA at instant k. Through the learning algorithm we ideally want that $\mu(k) \to x_o$ and $\sigma(k) \to 0$ as $k \to \infty$ where x_o is a maximum of f. However, for analytical tractability, we do not let $\sigma(k)$ to go to zero. We keep a parameter of the algorithm, σ_L, which is a suitably small positive constant, and set the goal of learning as $\mu(k) \to x_o$ and $\sigma(k)$ converging close to σ_L.

The system of automata considered in this paper is a general stochastic game with incomplete information played by a team of learning automata consisting of both FALA's and CALA's. In the theory of learning automata, algorithms for learning optimal strategies have been developed for many discrete and continuous games [2], [10]–[12] and have been used for adaptive decision making in many applications. However, there have been no results regarding algorithms for learning optimal strategies in a game consisting of both discrete and continuous parts.

This game model can be used, in general, for (locally) maximizing the empirical (or expected) risk in a concept learning problem where H is parameterized by some continuous and some discrete variables. We illustrate this for the case of learning simple conjunctive expressions over both nominal and linear attributes under classification noise. In this special case, it turns out that local maxima of the risk are all that we need to learn h^*.

In Section II, we formulate the hybrid game problem and define the notion of solutions of the game. We propose a decentralized learning algorithm and analyze its convergence properties in Section III. Section IV discusses a special case of the game where all players receive identical payoff. Section V presents an application of the game formulation to incremental learning of conjunctive concepts under classification noise. Section VI concludes the paper.

II. Problem Formulation

Consider a team of $N + M$ learning automata consisting of N finite action set learning automata (FALA) and M continuous action set learning automata (CALA) involved in a stochastic game with incomplete information. Let the action set of ith FALA be denoted by $\mathcal{S}_i$ with $|\mathcal{S}_i| = m_i$, $1 \le i \le N$. Let the jth CALA choose actions from the real line $\Re$, $1 \le j \le M$.

Let $a_i(k) \in \mathcal{S}_i$, denote the action chosen by the ith FALA, $i = 1, \cdots, N$, and let $x_j(k) \in \Re$, be the action chosen by the jth CALA, $j = 1, \cdots, M$, at the kth instant. We use $\mathbf{a}(k) = (a_1(k), a_2(k), \cdots, a_N(k))$ to denote the actions chosen by the FALA part of the team and $\mathbf{x}(k) = (x_1(k), x_2(k), \cdots, x_M(k))$, to denote the actions chosen by the CALA part of the team, at instant k. Let $Y(k) = (\mathbf{a}(k), \mathbf{x}(k))$, $Y \in (\prod_{i=1}^N \mathcal{S}_i) \times \Re^M \triangleq \mathcal{D}$. Each element of $\mathcal{D}$ is a tuple of action choices by the players, and, as per our notation, we denote an arbitrary element of $\mathcal{D}$ by $(\mathbf{a}, \mathbf{x})$. At each instant, after the selection of actions, the environment provides a stochastic reinforcement (also called payoff) to each of the automata. Let r_l be the payoff to the lth automaton player, $1 \le l \le N+M$. It is assumed that r_l takes values in [0, 1], for all l. Define functions $F^l\colon \mathcal{D} \to [0, 1]$, $1 \le l \le N+M$, by

$$F^l(\mathbf{a}, \mathbf{x}) = E[r_l | i\text{th FALA chose } a_i \text{ and } j\text{th CALA chose } x_j]. \tag{2.6}$$

F^l is called the payoff function for player l. The players only receive the payoff (or reinforcement) signal and they have no knowledge of the payoff functions.

Definition 2.1: We say $(\mathbf{a}^*, \mathbf{x}^*)$, $a^* = (a_1^*, \cdots, a_N^*)$; $\mathbf{x}^* = (x_1^*, \cdots, x_M^*)$, is an *optimal point* of the game if

1) For each i, $1 \le i \le N$,
$$F^i(\mathbf{a}^*, \mathbf{x}^*) \ge F^i(\mathbf{a}, \mathbf{x}^*),$$
for all $\mathbf{a} = (a_1^*, \cdots, a_{i-1}^*, a_i, a_{i+1}^*, \cdots, a_N^*)$ such that $a_i \ne a_i^*$, $a_i \in \mathcal{S}_i$.

2) For each j, $1 \le j \le M$, $\exists \epsilon > 0$ such that
$$F^{N+j}(\mathbf{a}^*, \mathbf{x}^*) \ge F^{N+j}(\mathbf{a}^*, \mathbf{x}),$$
for all $\mathbf{x}$ such that $\mathbf{x} \in \mathcal{B}^M(\mathbf{x}^*, \epsilon)$, where $\mathcal{B}^M(\mathbf{x}^*, \epsilon)$ is an ϵ-ball in $\Re^M$ centered at $\mathbf{x}^*$.

Remark 2.1: In the above definition, Condition 1) implies that $\mathbf{a}^*$ is a Nash equilibrium of the game matrix $F^l(\cdot, \mathbf{x}^*)$ indexed by a_i, $1 \le i \le N$. Condition 2) means that $\mathbf{x}^*$ is a local maximum of $F^l(\mathbf{a}^*, \cdot)$.

Now the learning problem is one of identifying optimal points of the game through repeated plays, that is, by repeatedly choosing actions and receiving respective payoffs. As outlined in Section I, for this, each automaton maintains a probability distribution over its action set which is modified after each play using a learning algorithm.

Let $\mathbf{p}_i(k) = [p_{i1}(k), \cdots, p_{im_i}(k)]$, $1 \le i \le N$, denote the action probability distribution of the ith FALA, where $p_{ij}(k) = \text{Prob}[a_i(k) = a_{ij}]$ and a_{ij} is the jth action in $\mathcal{S}_i$. The action probability distribution of jth CALA at kth instant is $N(\mu_j(k), \phi(\sigma_j(k)))$, which is normal distribution with mean $\mu_j(k)$ and standard deviation $\phi(\sigma_j(k))$ [the function $\phi(\cdot)$ is given by (2.9) below]. Let $\mathbf{c}_j(k) = [\mu_j(k), \sigma_j(k)] \in \Re^2$, $1 \le j \le M$. Then the *state* of ith FALA is given by $\mathbf{p}_i(k)$ and the *state* of jth CALA is given by $\mathbf{c}_j(k)$.

The state of the team, at instant k, is given by $S(k) = (P(k), C(k))$, where $P(k) = [\mathbf{p}_1(k), \cdots, \mathbf{p}_N(k)]$ and $C(k) = [\mathbf{c}_1(k), \cdots, \mathbf{c}_M(k)]$, $S(k) \in [0, 1]^{m_1+\cdots+m_N} \times \Re^{2M} \triangleq \mathcal{K}$. It may be noted that any point in $\mathcal{K}$ represents a probability distribution over $\mathcal{D}$, the set of all action tuples of the team.

At instant k, the ith FALA chooses $a_i(k) \in \mathcal{S}_i$ at random according to $\mathbf{p}_i(k)$, $1 \le i \le N$, and jth CALA chooses $x_j(k) \in \Re$ at random according to $N(\mu_j(k), \phi(\sigma_j(k)))$, $1 \le j \le M$. Then the lth player gets two reinforcements from the environment: $r_l(k)$ and $r'_l(k)$, $1 \le l \le M+N$. $r_l(k)$ is the response to the action tuple $(\mathbf{a}(k), \mathbf{x}(k))$ and $r'_l(k)$ is the response to the action tuple $(\mathbf{a}(k), \boldsymbol{\mu}(k))$.

Then each FALA updates its action probability distribution using the so called L_{R-I} learning algorithm as follows:

$$\mathbf{p}_i(k+1) = \mathbf{p}_i(k) + \lambda r'_i(k)[\mathbf{e}_{a_i} - \mathbf{p}_i(k)], \qquad 1 \le i \le N \tag{2.7}$$

where $\mathbf{e}_{a_i}$ is the unit vector of dimension m_i with a_ith component unity and $\lambda \in (0, 1)$ is the step size parameter of the algorithm.

The jth CALA updates its state as follows:

$$\begin{aligned}
&\mu_j(k+1)\\
&\quad = \mu_j(k) + \lambda \mathcal{F}_1(\mu_j(k), \sigma_j(k), x_j(k), r_{N+j}(k), r'_{N+j}(k))\\
&\sigma_j(k+1)\\
&\quad = \sigma_j(k) + \lambda \mathcal{F}_2(\mu_j(k), \sigma_j(k), x_j(k), r_{N+j}(k), r'_{N+j}(k))\\
&\qquad - \lambda K[\sigma_j - \sigma_L]
\end{aligned} \tag{2.8}$$

where $\mathcal{F}_1(\cdot)$, $\mathcal{F}_2(\cdot)$ are defined as

$$\mathcal{F}_1(\mu, \sigma, x, r, r') = \left(\frac{r - r'}{\phi(\sigma)}\right)\left(\frac{x-\mu}{\phi(\sigma)}\right)$$

$$\mathcal{F}_2(\mu, \sigma, x, r, r') = \left(\frac{r - r'}{\phi(\sigma)}\right)\left[\left(\frac{x-\mu}{\phi(\sigma)}\right)^2 - 1\right]$$

with

$$\phi(\sigma) = (\sigma - \sigma_L) I\{\sigma > \sigma_L\} + \sigma_L \tag{2.9}$$

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

Remark 2.2: The parameter σ_L above is a lower bound on the standard deviation of the normal distribution from which CALA choose actions. This is needed to ensure proper convergence of the algorithm.

Define g^l: $\mathcal{K} \to [0, 1]$, $1 \le l \le N + M$, by

$$\begin{aligned}
g^l(P, C) = E[r_l &| \text{state of } i\text{th FALA is } \mathbf{p}_i, 1 \le i \le N,\\
&\text{and the state of } j\text{th CALA is } \mathbf{c}_j, 1 \le j \le M]
\end{aligned} \tag{2.10}$$

where $P = (\mathbf{p}_1, \cdots, \mathbf{p}_N)$ and $C = (\mathbf{c}_1, \cdots, \mathbf{c}_M)$, $\mathbf{c}_i = (\mu_i, \sigma_i)$. Assuming $F^l(\mathbf{a}, \mathbf{x})$ is integrable, we can write, from (2.6)

$$g^l(P, C) = \int_{\Re^M} \sum_{j_1, \cdots, j_N} F^l(\mathbf{a}, \mathbf{x}) \prod_k p_{kj_k} dN(\boldsymbol{\mu}, \Sigma) \tag{2.11}$$

where $\mathbf{a} = (a_{j_1}, \cdots, a_{j_N})$, $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_M)$, Σ is the $M \times M$ diagonal matrix with the ith diagonal entry being $(\phi(\sigma_j))^2$. [As a notation we represent such diagonal matrices by diag(d_i), where d_i is the ith diagonal entry.] $N(\boldsymbol{\mu}, \Sigma)$ is the multidimensional Gaussian distribution with density

$$\frac{1}{|\Sigma|^{1/2}(2\pi)^{M/2}} \exp((\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})).$$

For convenience, we use C and $(\boldsymbol{\mu}, \Sigma)$ interchangeably.

The payoff functions $F^l(\cdot)$, $l = 1, \cdots, N+M$, are defined over the space of all tuples of actions (given by $\mathcal{D}$) and the optimal points of the game defined by Definition 2.1 are those tuples of actions which (in a special sense) "locally maximize" all payoff functions. However, since the automata learning algorithm searches over the space of probability distributions (represented by $\mathcal{K}$ defined earlier), the algorithm converges to a point in $\mathcal{K}$ to maximize $g^l(\cdot)$ functions defined over $\mathcal{K}$ by (2.11). In the following two definitions, we characterize some points of $\mathcal{K}$ as maximal and modal points which can be seen to be notions closely related to the optimal points of game.

Definition 2.2: We say $(P^*, C^*) \in \mathcal{K}$, $P^* = (\mathbf{p}_1^*, \cdots, \mathbf{p}_N^*)$, $C^* = (\boldsymbol{\mu}^*, \Sigma^*)$, is a *maximal point* of the game if

1) For each i, $1 \le i \le N$,

$$g^i(P^*, C^*) \ge g^i(P, C^*),$$

for all $P \in [0, 1]^{m_1 + \cdots + m_N}$ such that $P = [\mathbf{p}_1^*, \cdots, \mathbf{p}_{i-1}^*, \mathbf{p}_i, \mathbf{p}_{i+1}^*, \cdots, \mathbf{p}_N]$, $\mathbf{p}_i$ is a probability vector and $\mathbf{p}_i \ne \mathbf{p}_i^*$.

2) For each j, $1 \le j \le M$, $\exists \epsilon > 0$ such that

$$g^{N+j}(P^*, C^*) \ge g^{N+j}(P^*, (\boldsymbol{\mu}, \Sigma^*)),$$

for all $\boldsymbol{\mu}$ such that $\boldsymbol{\mu} \in \mathcal{B}^M(\boldsymbol{\mu}^*, \epsilon)$ where $\mathcal{B}^M(\boldsymbol{\mu}^*, \epsilon)$ is an ϵ-ball in $\Re^M$ centered at $\boldsymbol{\mu}^*$.

Define, for $P = [\mathbf{p}_1, \cdots, \mathbf{p}_N]$, $C = [\boldsymbol{\mu}, \Sigma]$, $\mathbf{x} \in \Re^M$, and $\mathbf{a} \in \prod_i \mathcal{S}_i$,

$$\begin{aligned}
H^l(P, \mathbf{x}) &\triangleq E[r_l | P, \text{CALA part chose } \mathbf{x}]\\
&= \sum_{j_1, \cdots, j_N} F^l((a_{j_1}, \cdots, a_{j_N}), \mathbf{x}) \prod_k p_{kj_k}
\end{aligned} \tag{2.12}$$

$$\begin{aligned}
J^l(\mathbf{a}, C) &\triangleq E[r_l | \text{FALA part chose } \mathbf{a}, C]\\
&= \int_{\Re^M} F^l(\mathbf{a}, \mathbf{x})\, dN(\boldsymbol{\mu}, \Sigma).
\end{aligned} \tag{2.13}$$

Then, we can write $g^l(\cdot, \cdot)$ given by (2.11) as

$$g^l(P, C) = \sum_{j_1, \cdots, j_M} J^l((a_{j_1}, \cdots, a_{j_N}), C) \prod_k p_{kj_k} \tag{2.14}$$

$$= \int_{\Re^M} H^l(P, \mathbf{x})\, dN(\boldsymbol{\mu}, \Sigma). \tag{2.15}$$

We can correspond each $(\mathbf{a}, C)$ with an $S = (P, C)$ where $P = [\mathbf{e}_{a_1}, \cdots, \mathbf{e}_{a_N}]$. We call such an S as a *corner* point of $\mathcal{K}$. It is easy to see that for each corner $S = (P, C)$ corresponded in this way with $(\mathbf{a}, C)$, we have $g^l(P, C) = J^l(\mathbf{a}, C)$.

Definition 2.3: We say $(\mathbf{a}^*, C^*)$, $a^* = (a_1^*, \cdots, a_N^*)$; $C^* = (\boldsymbol{\mu}^*, \Sigma^*)$, is a *modal point* of the game if

1) For each i, $1 \le i \le N$,

$$J^i(\mathbf{a}^*, C^*) \ge J^i(\mathbf{a}, C^*),$$

for all $\mathbf{a} = (a_1^*, \cdots, a_{i-1}^*, a_i, a_{i+1}^*, \cdots, a_N^*)$ such that $a_i \ne a_i^*$, $a_i \in \mathcal{S}_i$.

2) For each j, $1 \le j \le M$, $\exists \epsilon > 0$ such that

$$J^{N+j}(\mathbf{a}^*, C^*) \ge J^{N+j}(\mathbf{a}^*, (\boldsymbol{\mu}, \Sigma^*)),$$

for all $\boldsymbol{\mu}$ such that $\boldsymbol{\mu} \in \mathcal{B}^M(\boldsymbol{\mu}^*, \epsilon)$ where $\mathcal{B}^M(\boldsymbol{\mu}^*, \epsilon)$ is an ϵ-ball in $\Re^M$ centered at $\boldsymbol{\mu}^*$.

Remark 2.3: Suppose $(\mathbf{a}^*, C^*)$, $C^* = (\boldsymbol{\mu}^*, \Sigma^*)$, $\Sigma^* = \text{diag}(\phi(\sigma_i^*))$, is a modal point. Then, by (2.13), $(\mathbf{a}^*, \boldsymbol{\mu}^*)$ is arbitrarily close to an optimal point, if σ^* is sufficiently small. (This is intuitively clear because, as the variance decreases, the integral in (2.13) is essentially given by $F^l(\mathbf{a}, \boldsymbol{\mu})$. A formal proof follows easily by the same arguments as in [10, Lemma 2.4].)

Remark 2.4: If $(\mathbf{a}^*, C^*)$ is a modal point, we call the corresponding corner point (P^*, C^*), a *pure* maximal point.

Definition 2.4: (P^*, C^*) [resp. $(\mathbf{a}^*, C^*)$] is a strict maximal point (resp. strict modal point) if (P^*, C^*) [resp. $(\mathbf{a}^*, C^*)$] satisfies Definition 2.2 (resp. Definition 2.3) with strict inequalities for both conditions 1) and 2).

In the next section, we analyze the learning algorithm used by the hybrid team and show that the algorithm asymptotically identifies the modal points and consequently, by Remark 2.3, the optimal points of the game to a good approximation. Before that, we state some results which will be useful in the analysis to follow.

Define, for $P = (\mathbf{p}_1, \cdots, \mathbf{p}_N)$ and $C = (\mathbf{c}_1, \cdots, \mathbf{c}_M)$,

$$\begin{aligned} h_{lq}(P, C) =& E[r_l|\text{state of } i\text{th FALA is } \mathbf{p}_i,\ 1 \le i \le N,\\ & i \ne l,\ l\text{th FALA chooses action } q \text{ and the}\\ & \text{state of } j\text{th CALA is } \mathbf{c}_j,\ 1 \le j \le M]\\ =& \sum \Bigg\{ J^l((a_{j_1}, \cdots, a_{j_{l-1}}, q, a_{j_{l+1}}, \cdots, a_{j_N}),\\ & C) \prod_{k \ne l} p_{kj_k} \Bigg\} \end{aligned} \tag{2.16}$$

by (2.10) and (2.14). [The summation in (2.16) is over j_i, $1 \le i \le N$, $i \ne l$]. It can be noted from (2.14) and (2.16) that

$$g^l(P, C) = \sum_{q=1}^{m_l} h_{lq}(P, C) p_{lq}. \tag{2.17}$$

Lemma 2.1: (P^0, C^0) where $P^0 = [\mathbf{p}_1^0, \cdots, \mathbf{p}_N^0]$ and $C^0 = [\mathbf{c}_1^0, \cdots, \mathbf{c}_M^0]$, satisfies condition 1) of Definition 2.2 if and only if

$$h_{is}(P^0, C^0) \le g^i(P^0, C^0), \qquad 1 \le s \le m_i,\ 1 \le i \le N.$$

Corollary 2.2: Let (P^0, C^0) (as in Lemma 2.1) satisfy condition 1) of Definition 2.2. Then, for each i, $1 \le i \le N$,

$$h_{is}(P^0, C^0) = g^i(P^0, C^0), \qquad \forall s \text{ such that } p_{is}^0 > 0.$$

Both these results are just restatements of some of the properties of Nash equilibria [12], [13]. They can be proved directly from the definition and we skip the details.

III. Analysis of the Algorithm

Before we present a formal analysis of the algorithm described in the previous section, we first give a simple overview of our main results.

First consider a special case of the game where all players get the same payoff, that is, $F^l(\cdot, \cdot) = F(\cdot, \cdot)$, $\forall l$. This function, defined over $\mathcal{D}$, is a function of some continuous and some discrete variables. The optimal points of the game now correspond to "local" maxima of F. The game formulation presented in the previous section allows for a more general setting of multiple payoffs and the optimal points are like Nash equilibria. The goal of the automata algorithm is to identify these optimal points. For this, the automata approach is to search in the space of probability distributions over $\mathcal{D}$. It is easy to see that the *state* of the team at k, $S(k) = (P(k), C(k)) \in \mathcal{K}$, is a probability distribution over $\mathcal{D}$. Thus, the state of the team, $S(k)$, can converge (if at all) only to a point in $\mathcal{K}$. It is for this reason that we defined maximal and modal points. In view of Remarks 2.3 and 2.4, we want $S(k)$ to converge to some pure maximal point where the standard deviations of the normal distributions representing the action probability distributions of the CALA members of the team are sufficiently small. This is essentially what we prove in this section.

The analysis of the longtime behavior of $S(k)$ proceeds in two steps. The learning algorithm presented in the previous section specifies a stochastic difference equation which governs the evolution of $S(k)$. In the first step of the analysis we obtain an ordinary differential equation (ODE) that approximates this difference equation. This is a standard technique in analyzing such stochastic algorithms [8], [14]. The specific details of how these general techniques can be used to analyze automata algorithms are also known (see, e.g., [15, sec. 3]). Hence we simply state the result about the approximating ODE for our algorithm in Theorem 3.1.

The nature of our ODE approximation is such that the solution of the stochastic difference equation and that of the ODE would be arbitrarily close for an arbitrarily long time by taking the stepsize, λ, in the algorithm sufficiently small (see, e.g., [14, ch. 2, theor. 1] and [15, sec. 3]). Thus the second part of our analysis concentrates on characterizing the asymptotic solutions of the ODE. This is done in Theorem 3.2. Then, in Theorem 3.3, we provide a sufficient condition to ensure that the ODE solution converges to an equilibrium point rather than, e.g., exhibit a limit cycle behavior. It will be seen in the next section that this condition will always be satisfied for a game where all players get the same payoff. We make the following assumptions.

A1) For each player, conditioned on the action tuple chosen by the team, the reinforcement is independent of the past reinforcements.

A2) $F^l(\mathbf{a}, \mathbf{x})$ is continuously differentiable w.r.t. $\mathbf{x}$ for every $\mathbf{a}$, l.

A3) For every l and $\mathbf{a}$, $\exists$ a constant $B < \infty$ such that

$$\sup_x \|\nabla_{\mathbf{x}} F^l(\mathbf{a}, \mathbf{x})\| < B.$$

A4) For every l and $\mathbf{a}$, $F^l(\mathbf{a}, \cdot)$ has finite number of maxima in a compact set and has no maxima at infinity.

Remark 3.1: Assumptions A2)–A4) specify smoothness and growth conditions on the payoff functions. Since A2) holds, if $F^l(\mathbf{a}, \cdot)$ has compact support then A3) is automatically satisfied. These restrictions are required mainly to satisfy some integrability conditions. The class of functions under these assumptions still includes a large family of functions of interest.

We can write the state evolution of our algorithm as

$$S(k+1) = S(k) + \lambda G(S(k), \Psi(k))$$

where $\Psi(k) = (\mathbf{a}(k), \mathbf{x}(k), (r_1(k), \cdots, r_{N+M}(k)), (r_1'(k), \cdots, r_{N+M}'(k)))$, and $G(\cdot, \cdot)$ represents the updating given by (2.7) and (2.8).

Define the piecewise continuous interpolated version of $S(k)$ by

$$S^\lambda(t) = S(k), \qquad t \in [k\lambda, (k+1)\lambda),$$

where λ is the step size parameter for the learning algorithm [see (2.7) and (2.8)].

Theorem 3.1: Given the algorithm (2.7) and (2.8), the interpolated processes $S^\lambda(t)$ converge weakly as $\lambda \to 0$ to $S(.)$, the unique solution of the ODE

$$\frac{dS}{dt} = f(S), \qquad S(0) = S_0 \tag{3.18}$$

where

$$f(S) = E[G(S(k), \Psi(k)) | S(k) \equiv S]. \tag{3.19}$$

Proof: The proof follows easily by combining similar proofs from [10] and [12]. ■

As outlined at the beginning of this section, in the remaining part of the analysis we concentrate on characterizing the solutions of the ODE.

A. Analysis of the ODE

The equivalent ODE (3.18) of our algorithm will contain three sets of components corresponding to the $\mathbf{p}_i$'s, μ_j's, and σ_j's, respectively. We consider each set separately below.

1) The equations corresponding to $\mathbf{p}_i$'s are

$$\frac{dp_{iq}}{dt} = f_{iq}^p(S), \qquad 1 \le q \le m_i, \ 1 \le i \le N \tag{3.20}$$

where, from (2.7) and (3.19),

$$\begin{aligned} f_{iq}^p(S) &= p_{iq}(1-p_{iq})E[r_i' | S, a_i = q] \\ &\quad + \sum_{s \ne q} p_{is}(-p_{iq})E[r_i' | S, a_i = s], \\ &= p_{iq} \sum_s p_{is}[h_{iq}(S) - h_{is}(S)], \\ &\qquad \text{from (2.16).} \end{aligned} \tag{3.21}$$

2) The equations corresponding to μ_j's are

$$\frac{d\mu_j}{dt} = f_j^\mu(S), \qquad S = (P, C), 1 \le j \le M \tag{3.22}$$

where, using (2.8) and (3.19),

$$\begin{aligned} f_j^\mu(S) &= \int_{\Re^M} E\left[\left(\frac{r_{N+j} - r_{N+j}'}{\phi(\sigma_j)}\right) \middle| S, \mathbf{x}\right] \\ &\quad \cdot \left(\frac{x_j - \mu_j}{\phi(\sigma_j)}\right) dN(\boldsymbol{\mu}, \Sigma) \\ &= \int_{\Re^M} \left[\left(\frac{H^{N+j}(P, \mathbf{x})}{\phi(\sigma_j)}\right)\right] \\ &\quad \cdot \left(\frac{x_j - \mu_j}{\phi(\sigma_j)}\right) dN(\boldsymbol{\mu}, \Sigma) - 0, \\ &= \frac{\partial g^{N+j}}{\partial \mu_j}(S), \qquad \text{from (2.15).} \end{aligned} \tag{3.23}$$

Here we used (2.12) and the fact that $E[r_{N+j}'(k) | S, \mathbf{x}] = H^{N+j}(P, \boldsymbol{\mu})$, is not dependent on $\mathbf{x}$.

3) The equations corresponding to σ_j's are

$$\frac{d\sigma_j}{dt} = f_j^\sigma(S), \qquad S = (P, C), 1 \le j \le M \tag{3.24}$$

where, from (2.8) and (3.19),

$$\begin{aligned} f_j^\sigma(S) = E[(&\mathcal{F}_2(\mu_j(k), \sigma_j(k), x_j(k), r_{N+j}(k), r_{N+j}'(k)) \\ &- K[\sigma_j - \sigma_L]) | S]. \end{aligned}$$

We can write $f_j^\sigma(S) = \psi_j^\sigma(S) - K[\sigma_j - \sigma_L]$, where

$$\begin{aligned} \psi_j^\sigma(S) &= \int_{\Re^M} E\left[\left(\frac{r_{N+j} - r_{N+j}'(k)}{\phi(\sigma_j)}\right) \middle| S, \mathbf{x}\right] \\ &\quad \cdot \left[\left(\frac{x_j - \mu_j}{\phi(\sigma_j)}\right)^2 - 1\right] dN(\boldsymbol{\mu}, \Sigma) \\ &= \int_{\Re^M} \left[\left(\frac{H^{N+j}(P, \mathbf{x})}{\phi(\sigma_j)}\right)\right] \\ &\quad \cdot \left[\left(\frac{x_j - \mu_j}{\phi(\sigma_j)}\right)^2 - 1\right] dN(\boldsymbol{\mu}, \Sigma) - 0, \\ &= \frac{\partial g^{N+j}}{\partial \sigma_j}(S), \qquad \text{from (2.15).} \end{aligned}$$

Here we used (2.12) and the fact that $E[r_{N+j}'(k) | S, \mathbf{x}] = H^{N+j}(P, \boldsymbol{\mu})$, is not dependent on $\mathbf{x}$. Therefore

$$f_j^\sigma(S) = \frac{\partial g^{N+j}}{\partial \sigma_j}(S) - K[\sigma_j - \sigma_L]. \tag{3.25}$$

Define Γ_1, Γ_2, and Γ_3, subsets of $\mathcal{K}$, as

$$\Gamma_1 = \bigcap_{i,q} \{S \in \mathcal{K} | f_{iq}^p(S) = 0\} \tag{3.26}$$

$$\Gamma_2 = \bigcap_{j} \{S \in \mathcal{K} | f_{j}^\mu(S) = 0\} \tag{3.27}$$

$$\Gamma_3 = \bigcap_{j} \{S \in \mathcal{K} | f_{j}^\sigma(S) = 0\}. \tag{3.28}$$

That is, Γ_1, Γ_2, and Γ_3, respectively, denote the set of all equilibrium points of the three sets of ODE's given by (3.20), (3.22), and (3.24). The set of equilibrium points of ODE (3.18) is $\Gamma_1 \cap \Gamma_2 \cap \Gamma_3$.

Theorem 3.2: Let Γ_1, Γ_2, and Γ_3 be as defined by (3.26)–(3.28). Then

1) all corners of $\mathcal{K}$ belong to Γ_1;
2) all maximal points belong to $\Gamma_1 \cap \Gamma_2$;
3) an equilibrium point, $S^0 \in \Gamma_1 \cap \Gamma_2$, is unstable if it is not a maximal point;
4) all strict pure maximal points are asymptotically stable;
5) if $(P, (\boldsymbol{\mu}, \boldsymbol{\sigma})) \in \Gamma_3$, $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_M)$; $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_M)$, then each σ_j, $1 \le j \le M$, lies in a small neighborhood [determined by the parameter K of the learning algorithm (2.8)] around σ_L.

Proof:

1) Let $S^* = (P^*, C^*)$ be a corner of $\mathcal{K}$. We know that $P^* = [\mathbf{e}_{a_1}, \cdots, \mathbf{e}_{a_N}]$ for some $a_i \in S_i$, $1 \le i \le N$. Therefore, from (3.21), $f_{iq}^p(S^*) = 0$, $1 \le q \le m_i$, $1 \le i \le N$, since either $p_{iq}^* = 0$ or $p_{is}^* = 0$, $s \neq q$. Hence, $S^* \in \Gamma_1$.

2) Let $S^* = (P^*, C^*)$ be a maximal point. From (3.21),

$$\begin{aligned} f_{iq}^p(S^*) &= p_{iq}^* \sum_s p_{is}^*[h_{iq}(S^*) - h_{is}(S^*)], \\ &= p_{iq}^*[h_{iq}(S^*) - g^i(S^*)] \end{aligned} \tag{3.29}$$

by (2.17). Since S^* is a maximal point, it follows by Corollary 2.2 that $f_{iq}^p(S^*) = 0, 1 \le q \le m_i, 1 \le i \le N$, implying $S^* \in \Gamma_1$. Since S^* also satisfies condition 2) of Definition 2.2, it follows that $(\partial g^{N+j}/\partial \mu_j)(S^*) = 0$. Therefore, from (3.23), $f_j^\mu(S^*) = 0$, $1 \le j \le M$, implying $S^* \in \Gamma_2$. Hence, $S^* \in \Gamma_1 \cap \Gamma_2$.

3) Let $S^0 = (P^0, C^0)$ belonging to $\Gamma_1 \cap \Gamma_2$ not correspond to a maximal point. Therefore, at least one of the conditions of Definition 2.2 is not satisfied by S^0.

   a) Suppose condition 1) is not satisfied. Then, by Lemma 2.1, $\exists\, i, s$ such that $h_{is}(P^0, C^0) > g^i(P^0, C^0)$. By continuity of the functions involved, this inequality will hold for all points S in a small neighborhood around S^0. This implies, by (3.29), that for all points in this neighborhood, $(dp_{is}/dt)(S) = f_{is}^p(S) > 0$ if $p_{is} \neq 0$. Hence, S^0 is unstable.

   b) Suppose condition 2) is not satisfied. That is, $\boldsymbol{\mu}^0$ is not a local maximum of $g^{N+j}(P^0, \cdot)$ for some j. Since $d\mu_j/dt = \partial g^{N+j}/\partial \mu_j$ it is obvious that $\boldsymbol{\mu}^0$ will be unstable.

4) Let $S^0 = (P^0, C^0)$, $P^0 = [\mathbf{e}_{a_1^0}, \cdots, \mathbf{e}_{a_N^0}]$; $C^0 = (\boldsymbol{\mu}^0, \Sigma^0)$, be a strict pure maximal point. Then, we know that $Q^0 = (\mathbf{a}^0, C^0)$ is a strict modal point (cf. Remark 2.4). Let Q^0 satisfy condition 2) of Definition 2.3 with $\epsilon = \epsilon^0$. Define a region S_ρ by

$$\begin{aligned} S_\rho = \{(P, (\boldsymbol{\mu}, \Sigma^0)) \in \mathcal{K} | P = [\mathbf{p}_1, \cdots, \mathbf{p}_N], \\ \boldsymbol{\mu} \in \mathcal{B}^M(\boldsymbol{\mu}^0, \epsilon^0)\} \end{aligned}$$

where each $\mathbf{p}_i$, $1 \le i \le N$, is a probability vector of dimension m_i close to $\mathbf{e}_{a_i^0}$, and $\mathcal{B}^M(\mathbf{x}, \epsilon)$ denotes the open ball of radius ϵ centered at $\mathbf{x}$ in $\Re^M$.

Let $S = (P, C) \in S_\rho$. In $P = [\mathbf{p}_1, \cdots, \mathbf{p}_N]$, only $\sum_{i=1}^N (m_i - 1)$ components are independent since, for each i, $\sum_k p_{ik} = 1$. Choose p_{iq}, $q \neq a_i$, as the independent components. For these, we get

$$\frac{dp_{iq}}{dt} = H_{iq}(S) + \text{higher order terms in components of } P$$

where, by Taylor expansion of $h_{iq}(S)$ around S^0, by (2.16) and (3.21),

$$\begin{aligned} H_{iq}(S) = p_{iq}[J^i((a_1^0, \cdots, a_{i-1}^0, q, a_{i+1}^0, \cdots, a_N^0), C) \\ - J^i((a_1^0, \cdots, a_N^0), C)], \\ < 0, \qquad \forall q \neq a_i^0 \end{aligned} \tag{3.30}$$

since (P^0, C^0) is a strict pure maximal point. Define

$$\Phi(\sigma) = \frac{(\sigma - \sigma_L)^2}{2} I\{\sigma > \sigma_L\}, \qquad \sigma \in \Re. \tag{3.31}$$

Consider the Lyapunov function defined over S_ρ, for $S = (P, ((\mu_1, \cdots, \mu_N), \Sigma))$,

$$\begin{aligned} V(S) = \sum_{j=1}^M [g^{N+j}(P^0, C^0) - g^{N+j}(P^0, \hat{C}_j)] \\ + \sum_{j=1}^M K\Phi(\sigma_j) + \sum_{i,\, q:\, q \neq a_i} p_{iq}, \qquad S \in S_\rho \end{aligned}$$

where $\hat{C}_j = ((\mu_1^0, \cdots, \mu_{j-1}^0, \mu_j, \mu_{j+1}^0, \cdots, \mu_M^0), \Sigma^0)$. We have $V(S^0) = 0$ and $V(S) > 0$, $\forall S \in S_\rho$ since $S^0 = (P^0, C^0)$ is a strict pure maximal point.

$$\begin{aligned} \dot{V} = &-\sum_{j=1}^M \frac{\partial g^{N+j}}{\partial \mu_j} . \dot{\mu}_j - \sum_{j=1}^M \frac{\partial g^{N+j}}{\partial \sigma_j} . \dot{\sigma}_j \\ &+ \sum_{j=1}^M K[\sigma_j - \sigma_L]\dot{\sigma}_j + \sum_{i,\, q:\, q \neq a_i} \dot{p}_{iq} \\ = &-\sum_{j=1}^M (f_j^\mu)^2 - \sum_{j=1}^M (f_j^\sigma)^2 \\ &+ \sum_{i,\, q:\, q \neq a_i} p_{iq}[(J^i((a_1^0, \cdots, a_{i-1}^0, q, a_{i+1}^0, \cdots, a_N^0), \\ &\qquad C) - J^i((a_1^0, \cdots, a_N^0), C)) \\ &\qquad + \text{higher order terms}] \\ < &\, 0, \qquad \text{from (3.30).} \end{aligned}$$

Hence, S^0 is asymptotically stable.

5) Suppose $S = (P, (\boldsymbol{\mu}, \boldsymbol{\sigma})) \in \Gamma_3$, $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_M)$; $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_M)$. Then, $f_j^\sigma(S) = 0$, $\forall j$.

Let $\delta' > 0$. Take $K_j > (1/\delta') \sup_S |\partial g^{N+j}/\partial \sigma_j|$. We can always choose such a constant by assumption A3), since we can write, from (2.11),

$$\begin{aligned} \frac{\partial g^{N+j}}{\partial \sigma_j}(P, C) = \int_{\Re^M} \left(\frac{x_j - \mu_j}{\sigma_j}\right) \sum_{j_1, \cdots, j_M} \\ \cdot \frac{\partial F^{N+j}}{\partial x_j}(\mathbf{a}, \mathbf{x}) \prod_k p_{kj_k}\, dN(\boldsymbol{\mu}, \Sigma). \end{aligned}$$

Let $K = \max_j K_j$. If $\sigma_j < \sigma_L - \delta'$,

$$\begin{aligned} f_j^\sigma(S) &= \frac{\partial g^{N+j}}{\partial \sigma_j} - K_j[\sigma_j - \sigma_L] \\ &> \frac{\partial g^{N+j}}{\partial \sigma_j} + K\delta' \\ &> 0, \qquad \text{by choice of } K. \end{aligned}$$

If $\sigma_j > \sigma_L + \delta'$,

$$\begin{aligned} f_j^\sigma(S) &= \frac{\partial g^{N+j}}{\partial \sigma_j} - K_j[\sigma_j - \sigma_L] \\ &< \frac{\partial g^{N+j}}{\partial \sigma_j} - K\delta' \\ &< 0, \qquad \text{by choice of } K. \end{aligned}$$

Therefore, all zero crossings of $f_j^\sigma(S)$ have to occur within $(\sigma_L - \delta', \sigma_L + \delta')$. Now, it follows that all zeros of f^σ have to lie in an open ball of radius δ' centered at $\boldsymbol{\sigma}_L$. Hence, every point $(P, (\boldsymbol{\mu}, \boldsymbol{\sigma})) \in \Gamma_3$ is such that each component of $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_M)$ lies in a small neighborhood around σ_L. ■

Since the set of equilibrium points of the ODE (3.18) is $\Gamma_1 \cap \Gamma_2 \cap \Gamma_3$, in view of Theorems 3.1 and 3.2, we can conclude that the learning algorithm used by the team will not converge to a point in $\mathcal{K}$ that is not a maximal point, and all strict pure maximal points are locally asymptotically stable. Moreover, by Assertion 5 of Theorem 3.2, in any equilibrium point of the ODE (3.18), $(P, \boldsymbol{\mu}, \boldsymbol{\sigma})$, each component of $\boldsymbol{\sigma}$ lies in a small neighborhood around σ_L and so asymptotically the action chosen by each CALA will be close to the mean of the action probability distribution of the corresponding CALA (see Remark 2.3). If we had chosen the parameter σ_L of the algorithm to be sufficiently small, then the pure maximal point (P^0, C^0) that the algorithm converges to will be arbitrarily close to an optimal point of the game (cf. Definition 2.1). Still the theorem does not guarantee the convergence of the algorithm to a maximal point because the ODE may exhibit, e.g., limit cycle behavior. However, we give a sufficient condition under which the algorithm converges to a maximal point.

Theorem 3.3: Suppose there is a differentiable function Θ: $\Re^{m_1+\cdots+m_N+2M} \to \Re$, such that for some constants $b_1, b_2, b_3 > 0$

$$\frac{\partial \Theta}{\partial p_{iq}}(P, C) = -b_1 h_{iq}(P, C) \tag{3.32}$$

$$\frac{\partial \Theta}{\partial \mu_j}(P, C) = -b_2 \frac{\partial g^{N+j}}{\partial \mu_j}(P, C) \tag{3.33}$$

$$\frac{\partial \Theta}{\partial \sigma_j}(P, C) = -b_3\left[\frac{\partial g^{N+j}}{\partial \sigma_j}(P, C) - K[\sigma_j - \sigma_L]\right] \tag{3.34}$$

for all $(P, C) \in \mathcal{K} \subset \Re^{m_1+\cdots+m_N+2M}$. Let Θ be bounded below and suppose that there is a constant $c > 0$ such that

$$\Lambda_0 = \{(P, C) \in \mathcal{K} | \Theta(P, C) \le c\}$$

is a bounded set. Then, the automata team using the algorithm given by (2.7) and (2.8) converges to one of the maximal points, for any initial condition.

Proof: We have, from (3.20), (3.21), and (3.32)–(3.34),

$$\begin{aligned}\frac{d\Theta}{dt} &= \sum_{i,q} \frac{\partial \Theta}{\partial p_{iq}} \frac{dp_{iq}}{dt} + \sum_j \frac{\partial \Theta}{\partial \mu_j} \frac{d\mu_j}{dt} \\ &\quad + \sum_j \frac{\partial \Theta}{\partial \sigma_j} \frac{d\sigma_j}{dt} \qquad (3.35)\\ &= -b_1 \sum_{i,q} h_{iq} p_{iq} \sum_s p_{is}[h_{iq} - h_{is}] \\ &\quad - b_2 \sum_j \frac{\partial g^{N+j}}{\partial \mu_j} \frac{d\mu_j}{dt} \\ &\quad - b_3 \sum_j \frac{d\sigma_j}{dt}\left[\frac{\partial g^{N+j}}{\partial \sigma_j} - K[\sigma_j - \sigma_L]\right], \\ &= -b_1 \sum_i \sum_q \sum_{s>q} p_{iq} p_{is} [h_{iq} - h_{is}]^2 \\ &\quad - b_2 \sum_j (f_j^{\mu})^2 - b_3 \sum_j (f_j^{\sigma})^2 \le 0. \qquad (3.36)\end{aligned}$$

Thus Θ is nonincreasing along the trajectories of the ODE. Since Θ is bounded below and Λ_0 is a bounded set, by [16, Lemma 81, Ch. 4], asymptotically all the trajectories will be in the set $K_1 = \{(P, C) \in \mathcal{K} | (d\Theta/dt)(P, C) = 0\}$.

It is easy to see from (3.36) that if $(P^0, C^0) \in K_1$, then by (3.26)–(3.28), $(P^0, C^0) \in \Gamma_1 \cap \Gamma_2 \cap \Gamma_3$. That is, (P^0, C^0) is an equilibrium point of the ODE (3.18). Thus the ODE has to converge to some equilibrium point. Now the theorem follows by noting that all equilibrium points that are not maximal points are unstable by Theorem 3.2. ■

Theorems 3.1 and 3.2 characterize the asymptotic behavior of our algorithm. If, in addition, the sufficient conditions needed by Theorem 3.3 are met, the algorithm converges to one of the maximal points of the game. Of these, the pure maximal points are stable and strict pure maximal points are asymptotically stable, by Theorem 3.2. We cannot, in general, conclude anything about the stability of other maximal points. Since the L_{R-I} algorithm we use for FALA's has unit vectors as absorbing states, in practice, it is found that the algorithm converges to one of the pure maximal points.

IV. Special Case of Games with Common Payoff

In a game with common payoff, all the players receive the same payoff after each play. For this special case, we have $r_i = r_j$, and hence $F^i = F^j$, $\forall i, j$, where F^i is as given by (2.6). Hence the payoff structure of the game can be defined by a single function F: $\mathcal{K} \to \Re$

$$\begin{aligned}F(\mathbf{a}, \mathbf{x}) = E[r | i\text{th FALA chose } a_i \text{ and} \\ j\text{th CALA chose } x_j]. \qquad (4.37)\end{aligned}$$

Similarly we will have a single function $g(\cdot, \cdot)$ in place of $g^l(\cdot, \cdot)$ and a single function $J(\cdot, \cdot)$ in place of $J^l(\cdot, \cdot)$. In addition to the Assumptions A1)–A4) on $F(\cdot, \cdot)$, we assume the following:

A5) $F(\mathbf{a}, \cdot)$ vanishes outside a compact set in $\Re^M$.

Theorem 4.1: Consider a game played by a team of FALA's and CALA's with common payoff. Then the automata team using the algorithm given by (2.7) and (2.8) converges to one of the maximal points.

Proof: Define the function Θ over $\mathcal{K}$ by

$$\Theta(P, C) = -g(P, C) + \sum_{j=1}^{M} K\Phi(\sigma_j)$$

where $\Phi(\cdot)$ is as given by (3.31).

We have from (2.14), (2.16), and (2.17),

$$\frac{\partial \Theta}{\partial p_{iq}}(P, C) = -h_{iq}(P, C), \tag{4.38}$$

$$\frac{\partial \Theta}{\partial \mu_j} = -\frac{\partial g}{\partial \mu_j} \tag{4.39}$$

$$\frac{\partial \Theta}{\partial \sigma_j} = -\frac{\partial g}{\partial \sigma_j} + K[\sigma_j - \sigma_L]. \tag{4.40}$$

From Assumption A5) g vanishes outside a compact set. Since the second term in the definition of Θ is strictly convex, we can choose c (which is the constant in the definition of Λ_0 used in Theorem 3.3) to be the value of Θ just outside the support of g. Then the set Λ_0 of Theorem 3.3 is bounded. Now the proof follows by applying Theorem 3.3. ■

V. Application of the Hybrid Team to Concept Learning

In this section, we illustrate an application of the hybrid automata game described in Section IV to the problem of learning conjunctive concepts.[2]

The problem is to learn a concept in the form of a conjunctive logic expression (over a finite set of attributes) given a set of positive and negative examples. Some of the attributes could be nominal while others could be linear. Each nominal attribute assumes only finitely many values and each linear attribute takes values from a bounded interval in $\Re$. All examples will be described as tuples of values for these attributes and there may be classification noise in the training sample. Let the attributes chosen for the domain be Y_i that take values from sets V_i, $i = 1, \cdots, n$.

Definition 5.1: A concept description given by

$$\text{atom}_1 \wedge \cdots \wedge \text{atom}_s$$

is called a *simple conjunctive concept* where atom_i is a logic expression of the form $[Y_i \in v_i]$, $v_i \subset V_i$, and, further, if Y_i is a linear attribute then $v_i \subset \Re$ is a compact interval.

In the PAC learning framework described in Section I, the instance space of our problem, X, is $\prod_{i=1}^{n} V_i$; the outcome and decision spaces are $\{0, 1\}$; and the hypothesis space is the class of all simple conjunctive concepts. The labeled examples are drawn w.r.t. some distribution P over $X \times \{0, 1\}$. If we choose the loss function as $l(y, a) = I\{y = a\}$,[3] then the *correct* concept is given by

$$h^* = \arg\max_h E[l(y, h(x))]. \tag{5.41}$$

Now, to use the automata team algorithm for this problem we need to represent each concept with a pair $(\mathbf{a}, \mathbf{x})$ (where components of $\mathbf{a}$ come from discrete sets and $\mathbf{x}$ is a real vector). Then, given an example $(\mathbf{t}, y)$ (where $\mathbf{t}$ is a tuple of attribute values and y is the classification as given in the training sample) we can supply $l(y, h(\mathbf{t}))$ as the common reinforcement for the choice of action $(\mathbf{a}, \mathbf{x})$ (which corresponds to concept h). Thus $F(\cdot, \cdot)$ [defined by (4.37)] would become the expected value of l and the team can learn h^*. Hence, in designing an automata team to solve the concept learning problem we need to answer two questions: i) How do we formulate an automata team so that every tuple of actions of the team uniquely corresponds to an element of our concept space? ii) Since the automata team converges to optimal points of $F(., .)$, are there optimal points other than h^*? (It is easy to see that h^* would be an optimal point.)

[2] A detailed description of this algorithm for concept learning can be found in [17] where we also discussed how such teams of FALA and CALA can efficiently learn certain special classes of disjunctive concepts. But [17] does not contain proof of convergence for the algorithm, though the correctness of this approach for concept learning is established under the assumption that the automata team converges to the optimal points of the game.

[3] $I\{A\}$ is the indicator function of the event A.

Suppose the concept learning problem has N nominal attributes and M linear attributes. Every simple conjunctive concept is uniquely represented by a tuple $(v_1, \cdots, v_{N+M})$, $v_i \subset V_i$, $\forall i$. Further, for $N+1 \le i \le N+M$, the set v_i is an interval, say, $[c_i, d_i]$. Let n_i be the number of possible values for the ith nominal attribute. Then, given any conjunctive concept $(v_1, \cdots, v_{N+M})$, we can represent each v_i, for $1 \le i \le N$, by an n_i bit Boolean vector; and we can represent each v_i, for $N+1 \le i \le N+M$, by two real numbers. Hence, if we have an automata team with $2M$ CALA and $n_1 + \cdots + n_N$ FALA, where each FALA has two actions: {YES, NO}, then every tuple of actions of the team will represent a unique simple conjunctive concept.

We use this hybrid team of automata in a game with common payoff for learning the correct concept. At each instant each of the automata chooses an action. Recall from Section II that we need to generate two reinforcements at each instant for our learning algorithm. We generate these by checking whether or not the classification by the concept corresponding to the appropriate action tuple matches with that given in the training sample. For the reinforcement r the action tuple would be the actions chosen by the automata; and for r' the action tuple would consist of the actions chosen in case of FALA and the means of the action probability distributions in case of CALA. We shall refer to this algorithm as CLearn.

From the results of Section IV, we know that the automata team converges to one of the optimal points of the payoff function of this game (provided all the assumptions made in Section II hold for this game's payoff function, which is easily verified). Hence the next question to be answered is whether the optimal points of the payoff function are a good approximation to the correct concept h^*.

It is proved in [9] and [17] that (under some mild conditions on the probability distribution over our instance space and under uniform classification noise) the correct concept h^* would be an optimal point of the game and, further, any action tuple which does not correspond to h^*, would be an optimal point if and only if it corresponds to a simple conjunctive concept where the subset for one of the nominal attributes is null set. Since the correct concept will not contain null sets, correct learning for algorithm CLearn can be ensured by making it automatically loop back till the converged concept has no null sets though it cannot be proved that such a procedure always terminates. However, in our extensive empirical studies, algorithm CLearn always converged to correct concept and never needed such looping back.

The above result is true under any general classification noise. Specifically, let ν_z denote the probability with which the classification label of an instance z is corrupted before being given to the learning system. The above result that h^* is the only optimal point (modulo null sets) is still valid if $\nu_x < 0.5$, $\forall x$ [18]. However, for the case of more general noise (which includes combined attribute and classification noise) in the examples, all we can assert is that we converge to the optimal points of the payoff function which correspond to local maxima of the risk function as explained earlier. How good these local maxima will be as approximators to h^*, which is the global maximum, is dependent on the specific application.

TABLE I
PERFORMANCE OF ALGORITHM CLEARN ON WINE DATABASE

Noise %	λ	CPU Time Secs.	Error rate %
0	0.004	4	2.6
5	0.004	6	4.0
10	0.004	8	6.6
20	0.004	10	6.6
30	0.004	13	8.0
40	0.004	18	12.0

TABLE II
PERFORMANCE OF ALGORITHM DTREE ON WINE DATABASE

Noise %	Nodes	Average Depth	CPU Time Secs.	Error rate %
0	6	3.3	2.5	6.6
5	8	4.4	3.0	10.6
10	12	5.3	4.0	16.0
20	22	6.1	5.0	32.0
30	30	7.4	7.0	42.6
40	42	8.9	10.0	58.6

The algorithm CLearn that uses the hybrid team of automata model is a strictly incremental (or online) learning algorithm. It need not store any examples or any other detailed statistics of the set of examples. This is a distinct advantage compared to other PAC learning algorithms that can handle noisy examples. Further, our algorithm can be implemented in a distributed fashion.

A. Simulation Studies

The algorithm CLearn has been tested on many synthetic problems and also on many test problems from UCI machine learning (ML) database [19]. In this section, we present results obtained with the algorithm on one problem from the UCI ML database [19] and on one synthetic problem.

For comparison purposes, we have implemented a version of the decision tree based algorithm [20] that can handle both nominal and linear attributes effectively. We refer to this algorithm as Algorithm DTree.

Problem 1: The first problem we consider is the Wine recognition data from UCI ML database. These data are the results of a chemical analysis of wines grown in the same region but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines and these are provided as the values of 13 linear attributes. We consider the problem of classifying the first type of wine from the other two.

The domain consists of 178 examples. We divide the data into two sets of 103 and 75 examples and use the first one as training set and the second as test set. We study the performance of the algorithms for various classification noise rates by adding noise to the training set externally.

The results of simulation for Algorithm CLearn are given in Table I. The columns in the table indicate percent classification noise added to the training set, value of parameter λ in the algorithm, CPU time taken for learning and the final error rate on test set after learning. Table II contains the results for Algorithm DTree. In addition to noise, error rate and CPU time, Table II shows the size of the learnt decision tree in terms of the number of nodes and the average depth of tree.

Problem 2: In this problem, we consider a synthetic domain to illustrate the ability of algorithm CLearn to learn in the presence of both nominal and linear attributes. Let the domain be characterized by two nominal and two linear attributes. The nominal attributes Y_1 and Y_2 take values in $\{A, B, C, D\}$ and the linear attributes Y_3 and Y_4 from [0.0, 5.0]. Let the target conjunctive concept be

$$[\{Y_1 \in \{A, C\}\} \wedge \{Y_2 \in \{B, D\}\} \wedge \{Y_3 \in [2.0, 4.0]\} \\ \wedge \{Y_4 \in [2.0, 4.0]\}].$$

A fixed number of preclassified examples is generated randomly according to a predefined probability distribution for training the algorithms. For testing purposes, we generate another set of examples with respect to the same probability distribution.

The simulation results are presented below in a format similar to the previous problem. To bring out the advantages of incremental learning algorithms such as CLearn in contrast to the decision tree algorithms, we present simulations on two training sets of sizes 100 and 500 examples, respectively. In addition to showing the computational advantage of incremental learning, the results also show the generalization abilities of the algorithms.

Case 1: Training set size = 100; test set size = 100; results are in Tables III and IV.

Case 2: Training set size = 500; test set size = 100; results are in Tables V and VI.

TABLE III
PERFORMANCE OF ALGORITHM CLEARN ON PROBLEM 2: CASE 1

Noise %	λ	CPU Time Secs.	Error rate %
0	0.005	10	4.0
5	0.005	13	5.0
10	0.005	17	5.0
20	0.005	20	6.0
30	0.005	23	8.0
40	0.004	28	11.0

TABLE IV
PERFORMANCE OF ALGORITHM DTREE ON PROBLEM 2: CASE 1

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

TABLE V
PERFORMANCE OF ALGORITHM CLEARN ON PROBLEM 2: CASE 2

Noise %	λ	CPU Time Secs.	Error rate %
0	0.005	14	2.0
5	0.005	17	2.0
10	0.005	20	3.0
20	0.005	24	3.0
30	0.005	29	4.0
40	0.005	35	6.0

TABLE VI
PERFORMANCE OF ALGORITHM DTREE ON PROBLEM 2: CASE 2

Noise %	Nodes	Average Depth	CPU Time Secs.	Error rate %
0	15	4.6	2	3.0
5	17	5.4	5	4.0
10	24	6.8	18	8.0
20	39	7.3	28	14.0
30	55	8.2	44	22.0
40	61	8.9	58	28.0

From the simulation results, it is clear that the proposed automata based algorithm achieves good learning. The final error rate of the concept learnt is always smaller. This is particularly noticeable with noisy examples. The automata algorithm delivers an acceptable level of performance even under 40% classification noise while that of DTree deteriorates badly even with less than 20% noise. The time taken by the automata algorithm is more than that by DTree though the difference is not very large. One of the main computational overheads for CLearn is generation of normal random numbers. We were generating them as needed rather than storing a large file of numbers and reading from it. However, as the noise is increased, the time taken by Dtree becomes more. A feature to be noted here is that the automata algorithm is strictly incremental while DTree (and the other deterministic enumeration based optimization techniques used by other algorithms in computational learning theory) are not. Thus in problem 2, when the number of training samples is increased to 500 the time taken by CLearn shows only a small increase while that of DTree increases sharply. This problem would have been more severe for Dtree type algorithms if the number of attributes is increased. Since the automata algorithm is essentially parallel its time complexity would scale nearly linearly with increased number of attributes.

VI. DISCUSSION

In this paper we considered a stochastic game with incomplete information played by a hybrid team of learning automata. The team is called hybrid because some of the team members have finite action sets while others have continuous action sets. We defined optimal points for the game and proposed a distributed learning algorithm for learning these optimal strategies. In the common payoff case, the algorithm is a regression function learning technique for maximizing a function over continuous and discrete variables, based only on noisy observations of the function values at any chosen parameter settings.

The main motivation for considering this automata model is its relevance to concept learning with noisy examples. While learning concepts as logic expressions over nominal and linear attributes (a popular representation scheme for concept learning), this regression problem is difficult to solve because the concept space may have no simple algebraic structure on it and the loss function may be discontinuous. The strategy underlying the automata approach, namely, that of searching in the space of probability distributions over the concept space rather than searching the concept space directly, obviates the necessity of any algebraic structure on the concept space. Further, since the algorithm does not do any explicit gradient estimation (unlike, e.g., the stochastic approximation techniques), it would be numerically more robust.

Following the same procedure as in Section V, we can formulate the automata team to learn any general class of logic expressions. Even for general disjunctive concepts, such an automata team will learn a concept that is a local minimizer of the risk. How good such a concept would be as an approximator of h^* is application dependent.

It is also possible to use the general multiple payoff game model for concept learning. Here we would be giving different payoffs to different automata (or groups of automata). This, in general, means we use some features of the structure of the class of logic expressions chosen to solve the *credit assignment problem* more intelligently. For example, we can efficiently learn the so called k-term disjunctive expressions with *Marker* attributes under classification noise using such a multiple payoff game [9]. It is easy to see how this method can be used for other concept spaces also, e.g., halfspaces in $\Re^n$. In learning half spaces we need to learn a vector of real values, which can easily be done using a team of CALA only. And we can show, as here, that all local minima of the risk function correspond to the *correct* concept even under variable classification noise [18].

REFERENCES

[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[2] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[3] Y. Z. Tsypkin, *Adaptation and Learning in Automatic Systems*. New York: Academic, 1971.
[4] V. Vapnik, *Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1997.
[5] D. Haussler, "Decision theoretic generalizations of the PAC model for neural net and learning applications," *Inform. Comput.*, vol. 100, pp. 78–150, 1992.
[6] B. K. Natarajan, *Machine Learning: A Theoretical Approach*. San Mateo, CA: Morgan Kaufmann, 1991.
[7] M. Vidyasagar, *A Theory of Learning and Generalization with Applications to Neural Networks and Control Systems*. London, U.K.: Springer-Verlag, 1996.
[8] H. Y. Kushner and G. G. Yin, *Stochastic Approximation Algorithms and Applications*. New York: Springer-Verlag, 1997.
[9] K. Rajaraman, "Robust distribution free learning of logic expressions," Ph.D. dissertation, Dept. Elect. Eng., Indian Inst. Sci., Bangalore, India, 1996.

[10] G. Santharam, P. S. Sastry, and M. A. L. Thathachar, "Continuous action set learning automata for stochastic optimization," *J. Franklin Inst.*, vol. 331B, no. 5, pp. 607–628, 1994.

[11] M. A. L. Thathachar and P. S. Sastry, "Learning optimal discriminant functions through a cooperative game of automata," *IEEE Trans. Syst., Man, Cybern.*, vol. 17, no. 1, pp. 73–85, 1987.

[12] P. S. Sastry, V. V. Phansalkar, and M. A. L. Thathachar, "Decentralised learning of nash equilibria in multi-person stochastic games with incomplete information," *IEEE Trans. Syst., Man, Cybern,*, vol. 24, pp. 769–777, May 1994.

[13] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. New York: Academic, 1982.

[14] A. Benveniste, M. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations*. New York: Springer-Verlag, 1987.

[15] G. Santharam and P. S. Sastry, "A reinforcement learning neural network for adaptive control of Markov chains," *IEEE Trans. Syst., Man, Cybern. A*, vol. 27, pp. 588–601, Sept. 1997.

[16] M. Vidyasagar, *Nonlinear Systems Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[17] K. Rajaraman and P. S. Sastry, "A parallel stochastic algorithm for learning logic expressions under noise," *J. Ind. Inst. Sci.*, vol. 77, pp. 15–45, Jan. 1997.

[18] G. D. Nagendra, "PAC learning with noisy samples," M.E. thesis, Dept. Elect. Eng., Ind. Inst. Sci., Bangalore, India, Jan. 1997.

[19] P. M. Murphy and D. W. Aha, "UCI repository of machine learning databases," Dept. Inform. Comput. Sci., Univ. Calif., Irvine, 1994.

[20] U. M. Fayyad and K. B. Irani, "On the handling of continuous valued attributes in decision tree generation," *Mach. Learn.*, vol. 8, pp. 87–102, 1992.

K. Rajaraman received the B.E. degree in 1989 from Anna University, Madras, India, and the M.E. degree in 1991, and the Ph.D. degree in 1997, from the Indian Institute of Science, Bangalore, India.

He is presently on the Associate Research Staff at Kent Ridge Digital Labs, Singapore. His current research is focused on multilingual text mining, information retrieval, and machine learning.

P. S. Sastry received the B.Sc. (Hons.) degree in physics from the Indian Institute of Technology, Kharagpur, in 1978, and the B.E. degree in electrical communications engineering, the Ph.D. degree in electrical engineering from Indian Institute of Science, Bangalore, in 1981 and 1985, respectively.

He is currently Associate Professor, Department of Electrical Engineering, Indian Institute of Science, Bangalore. He has held visiting positions at the University of Michigan, Ann Arbor, and General Motors Research Labs, Warren, MI. His research interests include learning systems, pattern recognition, and computational neuroscience.