

- [15] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. New York: McGraw-Hill, 1987, pp. 547–553.
- [16] H. Zhuang, Z. S. Roth, and F. Hamano, "Observability issues in kinematic error parameter identification of manipulators," *ASME J. Dyn. Syst. Meas. Contr.*, vol. 114, pp. 319–322, June 1992.
- [17] *MACSYMA Reference Manual*, Symbolics, Inc., 1985.
- [18] H. O. Hartley, "The modified Gauss–Newton method for the fitting of nonlinear regression functions by least squares," *Technometrics*, vol. 3, no. 2, pp. 269–280, May 1961.
- [19] J. H. Borm and C. H. Menq, "Determination of optimal measurement configurations for robot calibration based on observability measure," *Int. J. Robotics Res.*, vol. 10, no. 1, pp. 51–63, Feb. 1991.

Learning Optimal Conjunctive Concepts through a Team of Stochastic Automata

P. S. Sastry, K. Rajaraman, and S. R. Ranjan

Abstract—The problem of learning conjunctive concepts from a series of positive and negative examples of the concept is considered. Employing a probabilistic structure on the domain, the goal of such inductive learning is precisely characterized. A parallel distributed stochastic algorithm is presented. It is proved that the algorithm will converge to the concept description with maximum probability of correct classification in the presence of up to 50% unbiased noise. A novel neural network structure that implements the learning algorithm is proposed. Through empirical studies it is seen that the algorithm is quite efficient for learning conjunctive concepts.

I. INTRODUCTION

This correspondence is concerned with a new class of algorithms for concept learning. Concept learning involves learning to classify objects belonging to a domain. To achieve this, the learning system will be supplied with preclassified examples (objects) from the domain. Any object of the domain characterizing the concept (that is, satisfying the "ideal" concept description) is called a positive example. All other objects from the domain are called negative examples. The goal of the learning system is a concept description that can be used in a decision procedure to decide whether or not a given domain object (positively) exemplifies the concept. The problem of learning to classify objects into different classes has been studied extensively in pattern recognition literature [14] as well. The main difference in concept learning is that the attributes used to describe different objects of the domain can take values in arbitrary sets and specifically attributes need not be numerical valued. Also, in concept learning, one is interested in obtaining concepts in a more human-comprehensible form (e.g., logic expressions that can be used in a rule for a decision) rather than in the form of values of parameters of an equation as in traditional statistical pattern recognition [6], [14].

This problem of inductive learning is studied extensively by psychologists [1] and computer scientists [2]–[4]. It is of interest in artificial intelligence (AI) due to the possibilities it offers in automating the knowledge acquisition process for expert systems (see

[3, ch. 4]). The problem of learning from examples is also of interest in the field of neural networks [4].

For the concept learning problem considered in this correspondence the domain is specified by giving a set of characteristic features called attributes. Each object of the domain, and hence in particular, each of the examples provided to the system is described by specific values assumed by the attributes. These attributes take values in arbitrary sets on which there may or may not be any structure. By experiencing a subset of such domain objects that are preclassified by a teacher, the system is to learn the "right" concept description.

In this correspondence, we view the concept learning problem in a probabilistic setting. Valiant is one of the first to realize the utility of such a point of view [5]. We see that in this formulation, we can objectively characterize what is meant by the "correct" concept description. We propose a stochastic algorithm based on a cooperating team of learning automata. This algorithm is shown to converge to the "correct" concept description for the class of problems where the correct concept can be expressed as a conjunctive concept (see [18] and [3, ch. 4]). The algorithm is very robust, and it is proved that it can tackle up to 50% of unbiased noise in examples. It is a parallel algorithm that learns incrementally and can be implemented on a parallel distributed network of simple processing elements with only local computation and hence can be viewed as a neural network, but in contrast with standard neural network models, here, the weights remain fixed, and the input-output functions of the neurons get adapted through the course of learning. It can possibly be called a dual neural network model.

II. CONCEPT LEARNING

In this correspondence, we assume that the objects of the domain are described by a set of features or attributes. Once the attributes are fixed, the form chosen for representing the concept will define the space of all possible concepts; the concept space. Some of the representations used in AI include logic expressions or rules and decision trees. In this correspondence, we use logic expressions. We can distinguish between at least two types of such rules (see [3, ch. 4]).

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

Definition 1: A concept description (logic expression) given by

$$[Y_1 \in v_1] \wedge \dots \wedge [Y_N \in v_N]$$

where v_i is a subset of V_i , $i = 1, \dots, N$, is said to be a Conjunctive expression.

Definition 2: A concept description of the form

$$C_1 \vee C_2 \vee \dots \vee C_m$$

where each C_i is a conjunctive expression is said to be in disjunctive normal form (DNF). We will refer to DNF concepts also as disjunctive expressions.

When we use conjunctive or disjunctive expressions to represent concepts, learning involves searching the space of all such rules to find the "best" rule given a set of examples. A rule from the concept space is said to be consistent with respect to a set of examples if it is satisfied by all the positive examples and is not satisfied by any negative example.

The goal of a concept learning algorithm is to identify the correct concept using a sequence of examples given to it. Hence, it is desirable to specify how one can be sure of correctness of such an algorithm. In general, this is a philosophical question of whether

Manuscript received May 24, 1991; revised October 17, 1992.

P. S. Sastry and K. Rajaraman are with the Department of Electrical Engineering, Indian Institute of Science, Bangalore, India.

S. R. Ranjan is with the Centre for Development of Advanced Computing, Pune University Campus, Pune, India.

IEEE Log Number 9207512.

induction based on a finite set of examples can ever be said to lead to provably correct generalizations. At a more mundane level, in the context of concept learning problems, two types of identification methods are distinguished in the literature: identification by enumeration and identification in the limit [2].

An algorithm that considers only those concepts that are consistent with all examples seen so far is employing identification by enumeration. Many AI algorithms achieve this identification. This method forces one to view any new example in the complete context of all examples seen so far. Since after seeing a finite set of examples the set of consistent concepts can still be very large, heuristic preferences for certain concept descriptions are made use of to reduce the search effort. If we do this pruning after seeing only some of the examples, then when more examples are presented, we may not be able to locate a consistent concept within the search space. This is one of the reasons why many AI algorithms tend to be nonincremental.

Identification in the limit views learning as an infinite process of making a series of guesses about the correct concept. Here, it is not required that all the guesses in the sequence be consistent. The eventual or limiting behavior of this sequence is what is used as a criterion of success. An algorithm is said to identify concept in the limit if after a finite (but not *a priori* bounded) number of examples are processed, the algorithm never changes its guess and that guess is consistent with all examples.

The identification criteria discussed above are useful only in noise-free environments. In a noisy environment, one cannot demand consistency with all the examples. In the presence of noise, the environment may present the learning system with two examples both described by same values for all attributes but one of them classified as positive and the other as negative by the teacher. There are three possible sources for such noise. First, the classes to be distinguished by the concept description may inherently be nonseparable under the chosen set of attributes. This would be the case, for example, in learning to diagnose a disease using a specific set of clinical tests. (The examples for learning in such a case can be generated by the respective physical processes to be distinguished.) Second, the measurement process that assigns values for attributes could introduce noise. Finally, the teacher who classifies the training examples may not be infallible. In any case, in the presence of noise, a positive example given to the algorithm may not, in fact, be a positive example in the sense that the "right" concept description should satisfy it. This is similar for negative examples. Such wrongly classified examples will be termed noisy examples, and they cannot be spotted *a priori* (i.e., before learning the right concept). While defining the criterion for "correct" concept, we need to make sure that the definition makes sense even in the presence of noise.

For a concept learning algorithm to be useful, the learnt concept description should not only be applicable to the examples provided by the teacher but also to other objects in the domain. To tackle this issue of generalization one has to have a way of distinguishing rules not only based on their performance on the presented examples but also based on their likely performance on the unseen objects of the domain. In many AI algorithms, this is handled, in a rather ad hoc manner, by using heuristic evaluation of the syntactic generality and simplicity of concept descriptions.

Both the issues of noise and generalization can be handled more rigorously if we assume a probabilistic structure on the domain, that is, if we assume the existence of probability distributions characterizing positive and negative examples. Valiant is the first to realize the utility of such a view point [5]. This idea is used in defining the notion of probably approximately correct (PAC) learning [18]. Our characterization of the objective of learning is similar to the concept of PAC learnability and is useful even in the presence of noise.

A. Problem Definition

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

1) A finite set of characteristic attributes of the domain are specified. Each attribute assumes only finitely many values; if it takes values in a continuum, then the range will be split into finitely many intervals. All examples will be described as tuples of values for these attributes. Also, it is assumed that the right concept can be described as a conjunctive logic expression involving a subset of these attributes. Nothing is known regarding the independence or otherwise of these attributes.

2) There exists a probability distribution (which is unknown) over the objects of the domain characterizing the concept.

Thus, it makes sense to ask the following: What is the probability that a specific description correctly classifies a random object from the domain? It is also assumed that the examples given to the learning system are randomly drawn for this distribution, but the probability distribution is totally unknown to the learning system. Thus, all that this assumption of a probabilistic structure amounts to is saying, in a precise sense, that the examples are representative of the concept.

Our algorithm makes a series of guesses about the right concept while performing a stochastic search over the concept space. As in identification in the limit, the criterion of success is the limiting behavior of the algorithm.

Let $J(\cdot)$ be a functional defined on the concept space. For any rule Z in the concept space, we define

$$J(Z) = \text{probability that } Z \text{ correctly classifies a random example from the domain.} \quad (1)$$

Let p_+ and p_- be the probabilities that a random object from the domain is a positive and a negative example, respectively.

Let us denote any object of the domain by $x = (x_1, \dots, x_N)$, where $x_i \in V_i$, is the value set of i th attribute V_i .

Let $P^+(x)$ and $P^-(x)$ denote the probabilities that the object x is a positive example and a negative example, respectively. Then, (1) can be rewritten as

$$J(z) = p_+ \sum P^+(x) \cdot I_z(x) + p_- \sum P^-(x) \cdot (1 - I_z(x)) \quad (2)$$

where $I_z(x)$ is 1 if the object x satisfies the logic expression Z and 0 otherwise. p_+ , p_- , $P^+(\cdot)$, and $P^-(\cdot)$ together characterize the probabilistic structure of the domain. All these quantities are unknown (it may be noted that $p_+ + p_- = 1$).

We define the correct concept to be the rule Z that globally maximizes $J(\cdot)$. However, it may be noted that given an x , we cannot compute $J(x)$ because the probability distributions in (2) are unknown.

It is easy to see that in the terminology of statistical pattern recognition, p_+ and p_- are prior probabilities, $P^+(\cdot)$ and $P^-(\cdot)$ are class conditional densities, and $J(Z)$ is probability of right classification with classifier Z . The main difference in concept learning is that elements of x (which is analogous to feature vector in pattern recognition) may not be numerical, and on the space to which x belongs, there may not be a meaningful distance measure. Thus, it would be difficult to use any standard pattern recognition algorithm for this problem.

In the next section, we present an algorithm based on a cooperating team of learning automata for the concept learning problem. Team of automata have been used for pattern recognition [15]. Though the model we use is different from that of [15], the main reason why automata can be used both for pattern recognition and concept

learning is that automata algorithms search in the space of probability distributions over the set of possible parameters rather than in the space of all parameters and, hence, do not need any distance measure on the space of parameters.

As mentioned earlier, in this correspondence, we prove the convergence of the algorithm only in problems when the correct concept (i.e., global maximum of $J(\cdot)$) is a conjunctive expression. Hence, we give some more definitions useful for conjunctive concepts.

From Definition 1, it is easy to see that a conjunctive concept can be represented by a tuple (v_1, v_2, \dots, v_N) , where v_i is a subset of V_i , which is the value set of i th attribute.

Definition 3: Let (v_1^*, \dots, v_N^*) be the correct concept. Then, v_i^* is called the correct set of the attribute $Y_i, i = 1, \dots, N$.

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 conjunctive concepts. The problem of learning DNF concepts with our method is addressed in [16].

III. A STOCHASTIC ALGORITHM FOR LEARNING CONJUNCTIVE CONCEPTS

In this section, we first introduce the concept of a cooperative game of automata. Then, we show how the model can be used for learning optimal conjunctive concepts in a noisy environment. Our treatment of learning automata will be brief. The reader is referred to [7] for further details.

A. Learning Automata

The learning automaton has a finite set of actions $A = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ from which it chooses an action at each instant. This choice is made randomly based on the so-called action probability distribution. Let $\mathbf{p} = [p_1(k), \dots, p_r(k)]$ denote this distribution. For the choice of action, the automaton gets a reaction (which is also called response or reinforcement) from the environment. This reaction is also stochastic whose expected value is d_i if the automaton has chosen $\alpha_i, d_i, i = 1, \dots, r$ are called reward probabilities of the environment and these are unknown to the automaton. The objective is for the automaton to learn the action with highest expected value of reinforcement. Let $d_m = \max \{d_i\}$. Then, α_m is called the optimal action, and we want $p_m(k)$ to go to unity asymptotically. For this, the automaton makes use of a learning algorithm (denoted by T), which updates the action probability distribution. Formally, $\mathbf{p}(k+1) = T(\mathbf{p}(k), \alpha(k), \beta(k))$, where $\alpha(k)$ is the action chosen at instant k , and $\beta(k)$ is the resulting reaction.

A learning algorithm we use later on, which is called the linear reward-inaction (L_{R-I}) algorithm [7], is described below.

1) *L_{R-I} Algorithm:* Let $\alpha(k) = \alpha_i$ and let $\beta(k)$ be the response obtained at k . Then, $\mathbf{p}(k)$ is updated as follows:

$$\begin{aligned} p_i(k+1) &= p_i(k) + \mu\beta(k)[1 - p_i(k)] \\ p_j(k+1) &= p_j(k) - \mu\beta(k)p_j(k), j \neq i \end{aligned} \quad (3)$$

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

2) *Games of Automata:* Games with incomplete information played by learning automata have been used as models for adaptive decentralized decision making [7]. In such a model, the automata correspond to the players and the actions of the automata to the various pure strategies available to the players. A play of the game consists of a choice of action by each automaton. The response to the automaton is the payoff to the corresponding player that is assumed to be stochastic. For the purposes of this correspondence, our interest is in cooperative games with common payoff.

Consider N automata in a cooperative game with common payoff. Let the i th automaton have r_i actions to choose from ($i = 1, \dots, N$). In each play, all the team members choose actions independently and randomly, depending on their current action probability vectors. The overall action selection gets a response from the environment that is the common reaction to all the automata. The reward structure of the game can be represented by a $r_1 \times r_2 \times \dots \times r_N$ dimensional hypermatrix, $D = [d_{i_1 i_2 \dots i_N}]$ defined by $d_{i_1 i_2 \dots i_N} = E[\text{Response}]_{j \text{th}}$ player uses strategy i_j .

D is called the reward matrix of the game.

If the choice of the strategies m_i by i th player ($i = 1, \dots, N$) such that

$$d_{m_1 m_2 \dots m_N} = \max_{i_1 i_2 \dots i_N} \{d_{i_1 i_2 \dots i_N}\}$$

then m_i is called the optimal strategy of player $i, (i = 1, \dots, N)$ and the overall strategy selection (m_1, \dots, m_N) is called the optimal set of strategies of the team.

Definition 4: Any entry, $d_{i_1 i_2 \dots i_N}$, of the N -dimensional matrix D is called a mode if

$$\begin{aligned} d_{i_1 i_2 \dots i_N} &= \max_{j_1} d_{j_1 i_2 \dots i_N} \\ &= \max_{j_2} d_{i_1 j_2 \dots i_N} \\ &\vdots \\ &= \max_{j_N} d_{i_1 i_2 \dots j_N}. \end{aligned}$$

Theorem 1: Let D be the reward matrix of a cooperative game with common payoff played by N automata. Let all the automata use identical L_{R-I} algorithms. Then asymptotically as $\mu \rightarrow 0$, the team converges weakly to one of the modes of the matrix D .

For unimodal game matrix, the above theorem is proved in [8]. Using weak convergence techniques, the result may be proved for multimodal matrices also (see, e.g., [17]).

B. Algorithm for Concept Learning:

We formulate the problem of concept learning as a cooperative game with common payoff played by a team of learning automata.

Our notation is as follows. The problem has N attributes Y_1, Y_2, \dots, Y_N . The attribute Y_i takes values in the set V_i and $V_i = \{x_{i1}, x_{i2}, \dots, x_{in_i}\}, i = 1, \dots, N$. We denote arbitrary subset of V_i by either v_i or A_i .

Our model consists of a team of $n_1 \times n_2 \times \dots \times n_N$ automata, $X_{ij} (j = 1, \dots, n_i; i = 1, \dots, N)$ involved in a game with common payoff. Each automaton has two actions: YES and NO. The automaton X_{ij} is concerned with the decision of whether X_{ij} (the j th possible value of the i th attribute) is in the correct set of Y_i . At each instant, each of the automata chooses an action independently and at random based on its action probabilities. It is easy to see that this choice of actions results in the selection of a conjunctive concept by the team. Let the actions chosen by X_{ij} be $\alpha_{ij}, (j = 1, \dots, n_i; i = 1, \dots, N)$. Then, the conjunctive concept selected is (v_1, \dots, v_N) , where $X_{ij} \in v_i$ iff $\alpha_{ij} = \text{YES} (j = 1, \dots, n_i; i = 1, \dots, N)$. The team will then classify the next example with the selected concept. The teacher supplies the team with a response of 1 if the classification agrees with that of the teacher and 0 otherwise. This will be the common payoff or the reaction to all the automata. The automata then update the action probabilities using the L_{R-I} algorithm (see (3)).

Let $p_{ij}(k)$ denote the probability with which X_{ij} will choose action YES at instant k . Then, the algorithm can be described as below. Initially, we set $p_{ij}(0) = 1/2$ for all i, j . At each instant each of the automata, X_{ij} simultaneously and independently chooses

actions at random based on its current action probabilities. The response to the team is

$$\begin{aligned} \beta &= 1, \text{ if the classification done} \\ &\quad \text{by the automata team} \\ &\quad \text{matches with the teacher's} \\ &\quad \text{classification} \\ &= 0 \text{ otherwise.} \end{aligned} \quad (4)$$

Then, each automaton X_{ij} updates its p_{ij} as follows:

$$\begin{aligned} p_{ij}(k+1) &= p_{ij}(k) + \mu\beta[1 - p_{ij}(k)] \\ &\quad \text{if } X_{ij} \text{ chooses action YES at } k \\ &= p_{ij}(k) - \mu\beta p_{ij}(k) \\ &\quad \text{if } X_{ij} \text{ chooses action NO at } k \end{aligned} \quad (5)$$

where $\mu \in (0, 1)$ is the learning parameter.

If all the action probability vectors converge to unit vectors, then we say that the team has converged to the corresponding concept. The next subsection presents the convergence analysis of the algorithm.

Before proceeding with the analysis, it may be noted that our formulation of the concept learning problem includes possibility of noisy samples or 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 is correct.

C. Analysis of the Learning Algorithm

The action chosen by the team of automata will be a tuple of $n_1 \times n_2 \times \dots \times n_N$ elements ($\alpha_{ij} : 1 \leq j \leq n_i, 1 \leq i \leq N$). Thus, the reward matrix of the games D will be a $\prod_{i=1}^N n_i$ -dimensional hyper-matrix of dimension $2 \times 2 \times \dots \times 2$. As a notation, let $i_{j\bar{m}} = i_{j1}i_{j2}\dots i_{jm}$. Then, any element of D can be represented as $d_{i_{1n_1}i_{2n_2}\dots i_{Nn_N}}$, where each i_{jk} is either 1 or 0, corresponding to the two actions (YES and NO, respectively) of the automaton.

It is clear that the first n_1 automata are learning correct set for attribute Y_1 and so on. Hence, for the purpose of analysis, it would be convenient to talk of D as indexed by N subscripts, where each subscript is a set of attribute values. Therefore, we refer to the reward matrix entries as

$$d_{A_1 \dots A_N}$$

which actually refers to the element

$$d_{i_{1n_1}i_{2n_2}\dots i_{Nn_N}}$$

where $i_{jk} = 1$ iff $x_{jk} \in A_j$.

In view of Theorem 1, we know that the team will converge to one of the modes of the matrix, and the goal of the analysis is to characterize the modes. Our analysis proceeds in two steps. First, we consider the effect of noise on the reward matrix. Since the reaction to the automata team depends on its classification agreeing with that of the teacher, the probability of the team getting a reward (i.e., $\beta = 1$) depends not only on the probability of the chosen concept correctly classifying a random example (determined by the unknown probabilities $p_+, p_-, P^+(\cdot), P^-(\cdot)$) but also on the probability of wrong classification by the teacher. Let D denote the reward matrix when there is no noise, and let $D(\sigma)$ be the reward matrix under $\sigma\%$ unbiased noise. Theorem 2 shows that an element of $D(\sigma)$ is a mode if and only if the corresponding element of D is a mode, if $\sigma < 50$. Thus, under the condition of less than 50% noise, to understand the asymptotic behavior of our algorithm, it is enough to consider the modes of D . Theorem 3 characterizes the modes of D , thus proving the convergence result for the algorithm.

Remark 1: To prove some element is a mode, it must be shown to be greater than or equal to all the adjacent elements (see Definition 4). Two elements of D are adjacent if the corresponding action combinations differ in the choice of action for only one automaton. Therefore, we may talk of the adjacent entries as resulting from flipping the action of the one automaton. According to our terminology, the action combinations are represented by the tuple of sets (A_1, \dots, A_N) . Hence, the flipping of the action of an automaton refers to addition of one value to one A_i or deletion of one value from one A_i .

Remark 2: Learning of a conjunctive concept is essentially learning the correct sets of all attributes (see Definition 3). There are two important special cases of correct sets—the null set and the set of all possible value of an attribute. Assuming that there exists at least one positive example of the concept, the null set cannot be the correct set of any attribute. The correct set of an irrelevant attribute will be the set containing the entire range of the attribute.

Theorem 2: The order relations between various entries in the reward matrix for the game do not change if 1) the teacher is unbiased, and 2) the probability of noisy classification by the teacher does not exceed 0.5.

Proof: The expected reward to the team of automata for a set of actions chosen by the automata in the team resulting in subsets A_i ($i = 1, \dots, N$) of the range of the attribute Y_i ($i = 1, \dots, N$) is given by

$$\begin{aligned} d_{A_1 \dots A_N} &= p_+ \cdot \text{Prob}[\text{on a random positive example} \\ &\quad \text{the teacher's classification agrees} \\ &\quad \text{with that by the team}] \\ &+ p_- \cdot \text{Prob}[\text{On a random negative example} \\ &\quad \text{the teacher's classification} \\ &\quad \text{agrees with that by the team}]. \end{aligned}$$

If the probability of classification error by the teacher is σ and the teacher is unbiased, then $\text{Prob}(\text{a random positive example is classified negative by the teacher}) = \text{Prob}(\text{a random negative example is classified positive by the teacher}) = \sigma$; $\text{Prob}(\text{a random positive example is classified positive by the teacher}) = \text{Prob}(\text{a random negative example is classified negative by the teacher}) = (1 - \sigma)$.

Therefore,

$$d_{A_1 \dots A_N} = (1 - 2\sigma)Z + \sigma \quad (6)$$

where Z is the reward probability under no noise.

Therefore, the order relation between any two entries in the reward matrix is maintained as in the case of non-noisy environments if

$$(1 - 2\sigma) > 0$$

i.e.,

$$\sigma < 0.5.$$

Thus, the theorem follows.

Theorem 3: Consider the automata game for concept learning with the reward structure as defined by (4) with no noise.

Then, the following are true for the reward matrix.

- 1) The element in the reward matrix corresponding to (A_1^*, \dots, A_N^*) , where A_i^* is the correct set of Y_i , is a mode of the matrix.
- 2) In all other modes, the set of values selected for at least one attribute is null set.

Proof: The proof is given in the Appendix.

Theorem 3 together with Theorems 1 and 2 completes proof for the convergence of our learning algorithm.

Remark 3: By Theorem 3, we know that the correct concept description is a mode. Further, all other possible modes of D assign null set as the correct set of at least one attribute. Since in a concept learning problem, the correct set for an attribute cannot be a null set, it is easy to check whether the automata team has converged to the correct description or not. If the correct set of any attribute is null in the converged concept, then we can rerun the algorithm with a different starting point (i.e., a different seed to the random number generator).

IV. DISCUSSION

The learning algorithm presented in the previous section is a parallel stochastic scheme for learning conjunctive concepts.

Here, learning proceeds through a stochastic search over the space of all concepts. The stochastic nature of the search makes the algorithm immune to noise and we have proved that the algorithm can handle up to 50% of unbiased noise. The algorithm is incremental, that is, it processes one example at a time and does not need to store all examples.

In our concept learning model, we have assumed a probabilistic structure over the domain to precisely characterize the goal of learning. Since the underlying probability distributions are completely unknown to the automata team, the assumption of probabilistic structure only amounts to saying, in a precise manner, that the presented examples are representative of the concept. This viewpoint has helped us characterize the generalization properties and prove correctness of our algorithm.

There are many algorithms proposed for concept learning in AI. Three of the widely studied algorithms are candidate elimination algorithm (see [3, ch. 6]), the Star methodology and the AQ and INDUCE series of algorithms (see [3], [10], and [6, ch. 4]), and the *ID3* algorithm (see [3, ch. 15]). All the algorithms essentially achieve identification by enumeration. They can come out with the "best" concept that is consistent with all the examples.

Mitchell's algorithm [3] learns conjunctive concepts in an incremental manner. By employing a partial order on the concept space, the algorithm can incrementally update the region in concept space that is consistent with all the examples seen so far, and in the absence of noise, the algorithm converges to the correct concept. It is not possible to handle noise in this framework. Also, the algorithm becomes inefficient if one cannot impose a good partial order on the concept space [18].

In general, if no structure is assumed on the concept space, then to search for a consistent concept one needs to remember all the examples. Thus, Michalski's algorithm and Quinlan's *ID3* are both nonincremental. Both these methods can learn disjunctive normal form concepts.

Under the Star methodology of Michalski, different types of attributes are distinguished based on the structure imposed on their value sets. If the value set has no structure, then the attribute is called nominal; if the value set is totally ordered, then it is called linear, and if the value set is partially ordered, then the attribute is called structural [18], [3]. Depending on the types of attributes, a number of syntactic rules for generalization and specialization of concepts are proposed. The learning proceeds by starting with some concept description and generalizing and specializing it using various rules and keeping the "best" among the consistent concepts so obtained. Selection of best concepts is done by heuristic evaluation functions that use some ad hoc measures such as simplicity of the concept descriptions, etc. Though this algorithm is found useful in many problems, unless good background knowledge is available about the domain, this process tends to be very inefficient, particularly when

there are a large number of examples. Further, there is no effective way to handle noisy examples.

Quinlan's *ID3* algorithm learns the concept in the form of a decision tree, which is equivalent to a DNF concept. This algorithm makes use of a heuristic based on information theoretic considerations and is efficient. The algorithm is nonincremental but can handle some amount of noise [9], but we do not know of any result that assures that correct generalizations are made even in the presence of noise.

Our main contention is that for a useful analysis of the learning algorithm, one needs to be able to give some sort of guarantee for correctness of generalizations that are learned. While inductions in general may not be amenable for correctness proofs, as shown by Valiant [5], for concept learning, the idea of a probabilistic structure on the domain is very useful to be able to prove the correctness of the learned concept.

There are other automata team models for concept learning [11], [12]. An earlier automata model used N -automata for N -attribute problem where actions of automata are subsets of value sets [12]. While this algorithm also has a similar convergence proof, it is inefficient because each automaton will have 2^N actions. For the algorithm presented in this correspondence, the attributes may not be statistically independent, but if the attributes are independent, then one can use a slightly different learning algorithm that gives faster convergence in simulations [11].

Another important aspect of our algorithm is that unlike many of the AI algorithms, it is parallel. All automata independently choose their actions based on their respective probability distributions, get a common response from the environment (which can be broadcast to all of them), and then independently update their action probabilities. There need be no explicit communication between the automata. Hence, on an SIMD machine with one processor per automaton (or a group of automata), we can achieve almost linear speedup.

Viewed as such a parallel network, this algorithm is somewhat similar to reinforcement learning in neural networks [13]. We explore this aspect of the algorithm in the next section.

V. THE AUTOMATA TEAM AS A PARALLEL DISTRIBUTED NETWORK FOR CONCEPT LEARNING

The learning problem we have considered here is also of interest in the field of artificial neural networks. Unlike the case of learning algorithms such as backpropagation [4], here the network is not told what the correct output should be. What is available is only a noisy response from the teacher as to the correctness of the classification by the system. This corresponds to a reinforcement learning problem [13]. Our automata model can be viewed as a network but with one significant difference. In a general neural network the connection weights are updated through the process of learning, but here, the weights remain fixed, and the activation functions of the units get adapted through learning.

To concretise the details, we show in Fig. 1 a neural network that is equivalent to the automata team algorithm. The example (Z_1, \dots, Z_N) is presented to the system, where Z_i is the value of i th attribute in the example. Layer 1 units (refer to Fig. 1) are input units that funnel the attribute values to units in layer 2. There are $n_1 + \dots + n_N$ units (same as the number of automata in the team) in layer 2. (Recall that i th attribute can have n_i possible values). Units in layer 2 are the only learning units in the network. Layer 3 has N units with interconnections as shown in Fig. 1. (The weight of each connection is shown on the corresponding arc). The output of each layer 3 unit is a logical AND of all its inputs (all of which will be binary, i.e., 1/0). The single unit in layer 4, whose output is the logical AND of all the outputs of layer 3 units, produces the output of the network. If this unit's output is 1, then the network

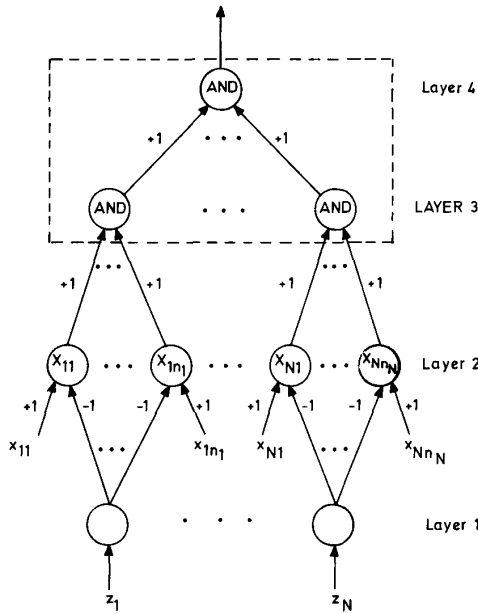


Fig. 1. Neural network implementation of the automata algorithm.

has classified the example as positive; otherwise, it is classified as negative. In this network, it is easy to see that we do not need layer 3 at all because the AND operation can be done by the layer 4 unit, but in the special case of this problem for independent attributes [11], we need the outputs of layer 3 units separately. Also, it is easier to understand the correspondence with automata team by keeping the layer 3 units. This is the reason why we show layers 3 and 4 in one dashed box in the figure.

All the connection weights (shown on the corresponding arcs in the figure) are fixed. The learning is effected by each of the layer 2 units learning the correct activation functions. Each unit in this layer X_{ij} ($j = 1, \dots, n_i; i = 1, \dots, N$) has two inputs. One is connected to the constant X_{ij} , which is the j th possible value of i th attribute, with weight $+1$, and the other is connected to Z_i , which is the output of i th unit in layer 1, with weight -1 . Thus, the net input to unit X_{ij} is $n_{ij} = X_{ij} - Z_i$. The output of X_{ij} will be $y_{ij} = h(n_{ij})$, where $h(\cdot)$ is the chosen activation function of the unit. The unit X_{ij} has two possible activation functions to choose from at each instant. They are $Ys(\cdot)$ and $Nt(\cdot)$ defined by

$$\begin{aligned} Ys(x) &= 1 \quad \forall x \\ Nt(x) &= 1 \quad \text{if } x \neq 0 \\ &= 0 \quad \text{if } x = 0. \end{aligned}$$

If X_{ij} has chosen the function $Nt(\cdot)$, then its output is 0 if and only if $X_{ij} = Z_i$. In all other cases $y_{ij} = 1$. It may be noted that $Z_i = X_{ij}$ for exactly one j . Thus the output of i th unit at layer 3 is 1 if and only if the unit X_{ij} , where j is such that $Z_i = X_{ij}$ has chosen the activation function $Ys(\cdot)$. Each unit X_{ij} functions as follows. At each instant k , it has an internal parameter $p_{ij}(k)$ and $p_{ij}(k) \in [0, 1]$ for all k . The unit X_{ij} generates a random number uniformly distributed over $[0, 1]$, and if this number is less than $p_{ij}(k)$, then at instant k , it chooses activation function $Ys(\cdot)$; otherwise, it chooses $Nt(\cdot)$ and produces its output accordingly.

For the classification done by the network, the environment or teacher gives a response $r, r \in \{0, 1\}$, which is broadcast to all the

layer 2 units. The learning algorithm for the network consists of all the layer 2 units updating their parameters p_{ij} by

$$\begin{aligned} p_{ij}(k+1) &= p_{ij}(k) + a[1 - p_{ij}(k)]r, \\ &\quad \text{if } X_{ij} \text{ chose } Ys(\cdot) \text{ at } k \\ p_{ij}(k+1) &= p_{ij}(k) - ap_{ij}(k)r, \\ &\quad \text{if } X_{ij} \text{ chose } Nt(\cdot) \text{ at } k \end{aligned} \quad (7)$$

where $a \in (0, 1)$ is a parameter.

Now, it is easy to see the correspondence between the automata team and the network. The unit X_{ij} corresponds to automaton X_{ij} of our team. The functions $Ys(\cdot)$ and $Nt(\cdot)$ correspond, respectively, to the two actions of the automaton YES and NO. p_{ij} corresponds to the probability of choosing YES for automaton X_{ij} . The learning algorithm given by (7) is same as the L_{R-I} algorithm given by (5). It is also clear that the analysis given in Section III is a convergence proof for this neural network. However, it must be said that this convergence proof was obtained only because we viewed the model as a game of learning automata and thus were able to use standard results from learning automata theory.

We can possibly term such neural network models as presented above dual neural networks because here the weights remain fixed, and the activation functions of units get adapted through learning. It is possible to view many learning automata games [7] as dual neural network algorithms. Formalization of general dual neural network models and examining whether one can get better learning algorithms for such networks are open problems worth investigation.

VI. SIMULATION RESULTS

In this section, we present the results of computer simulations of our learning algorithm (henceforth called Algorithm 1) on few synthetic domains. In synthetic problems, we first define a domain by specifying the attributes and their ranges. Then, an arbitrary concept description in conjunctive form involving the chosen attributes is selected. The training set of fixed number of classified examples of the concept is generated randomly according to a predefined probability distribution (under which the attributes are not stochastically independent). For each iteration of the automata algorithm, we select an example at random from this training set. In the example problems presented, both linear and nominal attributes have been considered. The performance of Algorithm 1 is studied under noise with probability of wrong classification by the teacher varied from 0 to 0.15.

An INDUCE type algorithm [10] (henceforth called Algorithm 2) was also implemented to compare its performance with our learning algorithm.

The results of the two algorithms on two synthetic problems are presented below. The results of algorithm 1 are given in Tables I–IV. In the tables, μ refers to the value of the learning coefficient used. The acronym WC refers to the number of runs in which the algorithm did not converge to the correct concept. The column percentage errors refers to the number of examples misclassified by the learned concept when tested on a new set of 100 examples. In the simulations of Algorithm 2, a training set was first chosen using the same procedure as for Algorithm 1. With this same training set, three runs of the algorithm were performed, each time changing only the order of the examples. The performance of Algorithm 2 is given in form of number of disjuncts in the learnt concept, together with the CPU time taken and percentage errors. (It may be noted that Algorithm 2 will learn, in general, a DNF description.) Algorithm 2 was simulated with zero noise only. The simulations were done on the SUN-3/60 Workstation.

TABLE I

Noise %	μ	Runs	WC	Avg. No. of Iterations	CPU times (mins)	Errors %
0	0.01	20	0	12500	1.37	0
2	0.01	20	0	17300	1.90	0
5	0.01	20	0	14300	1.57	0
8	0.01	20	2	14900	1.64	3
15	0.01	20	3	18400	2.02	6

TABLE II

Noise %	μ	Runs	WC	Avg. No. of Iterations	CPU Time (mins.)	Errors %
0	0.01	20	0	12800	1.40	0
2	0.01	20	0	14400	1.58	0
5	0.01	20	0	12300	1.35	0
8	0.01	20	1	10500	1.15	1
15	0.01	20	3	19800	2.17	2

TABLE III

Noise %	μ	Runs	WC	Avg. no. of Iterations	CPU Time (mins)	Errors %
0	0.008	20	2	24400	6.43	1
2	0.008	20	2	20500	4.80	1
5	0.008	20	1	25500	5.97	0
8	0.008	20	3	36800	8.65	2
15	0.008	20	3	43000	10.07	8

Example 1: Number of attributes = 4.

All attributes are nominal and assume values from $\{A, B, C, D\}$.
Description chosen:

$$\begin{aligned} & \{[ATT1 \in \{A, B\}] \wedge [ATT2 \in \{C, D\}] \\ & \quad \wedge [ATT3 \in \{A, B, C\}] \\ & \quad \wedge [ATT4 \in \{A, B, C, D\}]\}. \end{aligned}$$

Performance of Algorithm 1: Case 1: No. of examples = 100; $p_+ = 0.5$; see Table I.

Case 2: Number of examples = 200; $p_+ = 0.05$; See Table II.

Performance of Algorithm 2: Case a: No. of positive examples used = 50. No. of negative examples used = 50

- 1) Number of disjuncts = 1; Cpu time = 0.5 min; Errors = 0%;
- 2) Number of disjuncts = 4; Cpu time = 4.3 min; Errors = 2%;
- 3) Number of disjuncts = 3; Cpu time = 3.0 min; Errors = 2%.

Case b: No. of positive examples used = 100. No. of negative examples used = 100.

- 1) Number of disjuncts = 2; Cpu time = 2.2 mins; Errors = 0%
- 2) Number of disjuncts = 3; Cpu time = 3.4 mins; Errors = 2%
- 3) Number of disjuncts = 5; Cpu time = 5.5 mins; Errors = 4%

Example 2: No. of attributes = 4. The first attribute ATT1 is linear with range [0.0, 5.0]. All the other attributes are nominal and assume values from $\{A, B, C, D\}$.

Description Chosen:

$$\begin{aligned} & \{[ATT1 \in (0.5, 1.5) \cup (2.0, 3.5)] \\ & \quad \wedge [ATT2 \in \{A, B\}] \wedge [ATT3 \in \{C, D\}] \\ & \quad \wedge [ATT4 \in \{A, B, C\}]\}. \end{aligned}$$

Performance of Algorithm 1

Case 1: No. of examples = 100; $p_+ = 0.5$; see Table III.

Case 2: Number of examples = 200; $p_+ = 0.5$; see Table IV.

TABLE IV

Noise %	μ	Runs	WC	Avg. no. of Iterations	CPU Time (mins)	Errors %
0	0.008	20	0	25600	6.78	0
2	0.008	20	0	32100	7.52	0
5	0.008	20	1	30200	7.07	1
8	0.008	20	1	38000	8.90	2
15	0.008	20	2	42000	9.80	1

Performance of Algorithm 2

Case a: No. of positive examples used = 50. No. of negative examples used = 50.

- 1) Number of disjuncts = 6; Cpu time = 11.1 mins; Errors = 15%
- 2) Number of disjuncts = 5; Cpu time = 10.0 mins; Errors = 12%
- 3) Number of disjuncts = 8; Cpu time = 13.5 mins; Errors = 22%.

Case b:

Number of positive examples used = 100. No. of negative examples used = 100.

- 1) Number of disjuncts = 10; Cpu time = 20.5 mins; Errors = 18%
- 2) Number of disjuncts = 6; Cpu time = 12.0 mins; Errors = 20%
- 3) Number of disjuncts = 9; Cpu time = 16.3 mins; Errors = 14%.

It is observed that Algorithm 2 depended primarily on the sequence in which examples were presented from the training set. In the simulations, it learned different descriptions in different runs; the training set being the same. Also, most of the time it learned a big DNF expression even though the correct concept is a conjunctive expression. Therefore, just the availability of a representative set of examples does not guarantee correct convergence of Algorithm 2 unless they are in proper sequence. This is because of the non-incremental nature of Algorithm 2. On the other hand, we can except Algorithm 1 to converge to the correct concept independent of the order of presentation of examples, as the learning parameter $\mu \rightarrow 0$. This feature also makes our learning algorithm to exhibit generalization capabilities. In the simulations (see Table I and II and III and IV), when the number of examples in the training set was increased from 100 to 200, significant reduction in classification errors was observed unlike the case of Algorithm 2. Another feature that can be observed from the simulation runs is that the time taken by the automata algorithm does not depend on how many examples are there in the training set. On the other hand, for the same problem with more training set examples, Algorithm 2 takes more time.

VII. CONCLUSION

In this correspondence, we have considered the problem of learning conjunctive concepts using a set of positive and negative examples of the concept. We have posed this problem as a game played by a team of learning automata. Based on this model, a parallel stochastic learning algorithm is presented. We have proved that the algorithm will converge to the concept description having the maximum probability of correct classification in the presence of up to 50% of unbiased noise. We have also proposed a novel neural network structure that implements this algorithm. From the simulation studies, it is observed that the algorithm is quite efficient for learning conjunctive concepts.

One extension of the work outlined in this correspondence would be to study learning DNF concepts. This could be done by considering many teams of automata with each team learning a conjunctive expression [16]. Another extension would be to relax the condition

imposed in Section II-A for learning linear attributes so that "true" linear attributes could be learned. These two extensions form part of our future work.

APPENDIX

We give here the proof of Theorem 3.
The notation used is as follows.

- A_i the set $\{x_{ij} | x_{ij} \in V_i\}$ and automaton x_{ij} chooses YES
- A_i^* the correct set of i th attribute
- δd_i^1 incremental reward when a correct set value is added to A_i . (i.e., $d_{A_1 \dots A_i' \dots A_N} - d_{A_1 \dots A_i \dots A_N}$, where A_i' is obtained by adding a correct set value not present in A_i to A_i .)
- δd_i^2 incremental reward when a correct set value is removed from A_i
- δd_i^3 incremental reward when a wrong value (i.e., a value not present in the correct set) is added to A_i
- δd_i^4 incremental reward when a wrong value is removed from A_i
- p_+ probability of getting a +ve example
- p_- probability of getting a -ve example
- $P^+(\cdot)$ distribution from which the +ve examples are drawn
- $P^-(\cdot)$ distribution from which the -ve examples are drawn.

Proof of Theorem 3: The expected reward to the team of automata for a set of actions (which is chosen by automata in the team) resulting in subsets, A_i ($i = 1, \dots, N$) of the range of the attributes is given by

$$\begin{aligned}
 d_{A_1 A_2 \dots A_N} &= p_+ \cdot \text{Prob} \\
 &\quad (\text{a random +ve example is classified} \\
 &\quad \text{+ve by the team}) \\
 &+ p_- \cdot \text{Prob} (\text{a random} \\
 &\quad \text{-ve example is classified} \\
 &\quad \text{-ve by the team}) \\
 &= p_+ \cdot \left[\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \dots \right. \\
 &\quad \left. \sum_{k_N=1}^{n_N} P^+(x_{1k_1} \& \dots \& x_{Nk_N}) \right. \\
 &\quad \left. \cdot \prod_{j=1}^N I_{A_j}(x_{jk_j}) \right] \\
 &+ p_- \cdot \left[\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \dots \right. \\
 &\quad \left. \sum_{k_N=1}^{n_N} P^-(x_{1k_1} \& \dots \& x_{Nk_N}) \right. \\
 &\quad \left. \cdot \left\{ 1 - \prod_{j=1}^N I_{A_j}(x_{jk_j}) \right\} \right].
 \end{aligned}$$

Let A_i^* be the correct set for i th attribute $i = 1, \dots, N$. To prove

the theorem, we have to show that

- I) (A_1^*, \dots, A_N^*) is a mode,
- II) (A_1, \dots, A_N) is not a mode if no A_i is null, and A_i does not equal A_i^* at least for one i .

Part I: As explained earlier (Remark 1), we need to show that

$$d_{A_1^* \dots A_N^*} > d_{A_1^* \dots A_{i-1}^* A_i' A_{i+1}^* \dots A_N^*}, \forall i$$

where A_i' is different from A_i^* in exactly one element, i.e., A_i' either contains one extra element over A_i^* or all elements of A_i^* except one.

Case 1: Let $A_i' = A_i^* - \{x_{ic}\}$ for some x_{ic} belonging to A_i^* .

$$\begin{aligned}
 \delta d_i^2 &= d_{A_1^* \dots A_i' \dots A_N^*} - d_{A_1^* \dots A_i^* \dots A_N^*} \\
 &= p_+ \cdot \left[\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \dots \right. \\
 &\quad \left. \sum_{k_N=1}^{n_N} P^+(x_{1k_1} \& \dots \& x_{Nk_N}) \right. \\
 &\quad \left. \cdot \{I_{A_i'}(x_{ik_i}) - I_{A_i^*}(x_{ik_i})\} \right. \\
 &\quad \left. \cdot \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j^*}(x_{jk_j}) \right] \\
 &- p_- \cdot \left[\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \dots \right. \\
 &\quad \left. \sum_{k_N=1}^{n_N} P^-(x_{1k_1} \& \dots \& x_{Nk_N}) \right. \\
 &\quad \left. \cdot \{I_{A_i'}(x_{ik_i}) - I_{A_i^*}(x_{ik_i})\} \right. \\
 &\quad \left. \cdot \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j^*}(x_{jk_j}) \right].
 \end{aligned}$$

$P^-(\cdot)$ will be zero unless there is at least one x_{ik_i} not in its correct set. Since, A_j^* ($j = 1, \dots, i-1, i+1, \dots, N$) and A_i' contain only correct set values, $I_{A_j^*}(x_{jk_j})$ is zero if x_{jk_j} is not in its correct set and similarly for $I_{A_i'}(x_{ik_i})$. Hence, the second term in the above expression goes to zero.

Since, A_i' and A_i^* differ only in x_{ic} and $I_{A_i^*}(x_{ic}) = 1$; $I_{A_i'}(x_{ic}) = 0$, we have

$$\begin{aligned}
 \delta d_i^2 &= -p_+ \cdot \left[\sum_{k_1=1}^{n_1} \dots \sum_{k_{i-1}=1}^{n_{i-1}} \sum_{k_{i+1}=1}^{n_{i+1}} \dots \sum_{k_N=1}^{n_N} \right. \\
 &\quad \left. \cdot P^+(x_{1k_1} \& \dots \& x_{ic} \& \dots \& x_{Nk_N}) \right. \\
 &\quad \left. \cdot \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j^*}(x_{jk_j}) \right] < 0.
 \end{aligned}$$

Hence,

$$d_{A_1^* \dots A_N^*} > d_{A_1^* \dots A_i' \dots A_N^*}$$

if A_i' contains all but one element of A_i^* .

Case 2: Let $A_i' = A_i^* \cup \{x_{iw}\}$ for some x_{iw} not belonging to A_i^* . Proceeding as in Case 1 and noting that $I_{A_i^*}(x) = I_{A_i'}(x)$ for all

$x \neq x_{iw}$ and $I_{A_i^*}(x_{iw}) = 0$ & $I_{A_i'}(x_{iw}) = 1$, we get

$$\delta d_i^3 = -p_- \cdot \left[\sum_{k_1=1}^{n_1} \cdots \sum_{k_{i-1}=1}^{n_{i-1}} \sum_{k_{i+1}=1}^{n_{i+1}} \cdots \sum_{k_N=1}^{n_N} \right. \\ \cdot P^-(x_{1k_1} \& \cdots \& x_{iw} \& \cdots \& x_{Nk_N}) \\ \cdot \left. \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j^*}(x_{jk_j}) \right] < 0.$$

The above two cases are all the ways of changing A_i^* to A_i' . This is because A_i^* is a correct set, and the other two cases ($\delta d_i^1, \delta d_i^4$ mentioned in the notation) corresponding to adding a correct set value and deleting a wrong value are inapplicable. Hence, the concept formed by the correct sets of all the attributes is a mode.

This completes the proof that (A_1^*, \dots, A_N^*) is a mode.

Part II: To show (A_1, \dots, A_N) is not a mode when A_i are such that 1) $A_i \neq A_i^*$ for at least one i and 2) $A_i \neq \text{null}$, $\forall i$.

Choose an A_i such that $A_i \neq A_i^*$.

We know that $A_i \neq \emptyset$. Therefore, there are only two possibilities.

- 1) There is an $x_{iw}, x_{iw} \in A_i$ but $x_{iw} \notin A_i^*$.
- 2) There is an $x_{ic}, x_{ic} \notin A_i$, but $x_{ic} \in A_i^*$.

To show that (A_1, \dots, A_N) is not a mode, all we need is to show that there exists a A_i' differing from A_i in one element such that $d_{A_1 \dots A_i' \dots A_N} - d_{A_1 \dots A_N} > 0$.

It may be noted that we need not have to show the above inequality for all possible A_i' but for at least one A_i' .

Case 1: Let there be an $x_{iw}, x_{iw} \in A_i$ but $x_{iw} \notin A_i^*$.

Construct A_i' by deleting x_{iw} from A_i . As in Part I of the proof, the incremental reward in this case is given by

$$\delta d_i^4 = p_- \cdot \left[\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \cdots \sum_{k_N=1}^{n_N} P^-(x_{1k_1} \& \cdots \& x_{Nk_N}) \right. \\ \cdot \{I_{A_i}(x_{ik_i}) - I_{A_i'}(x_{ik_i})\} \\ \cdot \left. \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j}(x_{jk_j}) \right] \\ \delta d_i^4 = p_- \cdot \left[\sum_{k_1=1}^{n_1} \cdots \sum_{k_{i-1}=1}^{n_{i-1}} \sum_{k_{i+1}=1}^{n_{i+1}} \cdots \sum_{k_N=1}^{n_N} \right. \\ \cdot P^-(x_{1k_1} \& \cdots \& x_{iw} \& \cdots \& x_{Nk_N}) \\ \cdot \left. \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j}(x_{jk_j}) \right] > 0.$$

Case 2: From Case 1, we see that whenever a wrong value is present in any of the A_i , we can obtain an increment in reward by deleting that wrong value. Hence, we need to consider only those cases in which the A_i are such that there is no x_{iw} satisfying Case 1.

Let there be an $x_{ic}, x_{ic} \notin A_i$ but $x_{ic} \in A_i^*$. Such an x_{ic} exists, for otherwise, in view of the comments above, we must have $A_i = A_i^*$.

Construct A_i' by adding x_{ic} to A_i . The incremental reward is given by

$$\delta d_i^1 = p_+ \cdot \left[\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \cdots \sum_{k_N=1}^{n_N} P^+(x_{1k_1} \& \cdots \& x_{Nk_N}) \right. \\ \cdot \{I_{A_i'}(x_{ik_i}) - I_{A_i}(x_{ik_i})\} \\ \cdot \left. \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j}(x_{jk_j}) \right] \\ + p_- \cdot \left[\sum_{k_1=1}^{n_1} \sum_{k_2=1}^{n_2} \cdots \sum_{k_N=1}^{n_N} P^-(x_{1k_1} \& \cdots \& x_{Nk_N}) \right. \\ \cdot \{I_{A_i}(x_{ik_i}) - I_{A_i'}(x_{ik_i})\} \\ \cdot \left. \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j}(x_{jk_j}) \right].$$

The second term goes to zero by the same reason as given in Case 1 of Part I of the proof because A_i' contains only correct set values. Hence

$$\delta d_i^1 = p_+ \cdot \left[\sum_{k_1=1}^{n_1} \cdots \sum_{k_{i-1}=1}^{n_{i-1}} \sum_{k_{i+1}=1}^{n_{i+1}} \cdots \sum_{k_N=1}^{n_N} \right. \\ \cdot P^+(x_{1k_1} \& \cdots \& x_{ic} \& \cdots \& x_{Nk_N}) \\ \cdot \left. \prod_{\substack{j=1 \\ j \neq i}}^N I_{A_j}(x_{jk_j}) \right] > 0.$$

This completes Part II of the proof.

REFERENCES

- [1] E. B. Hunt, *Concept Learning: An Information Processing Problem*. New York: Wiley, 1973.
- [2] Angluin and Smith, "Inductive inference: Theory and methods," *ACM Comput. Surveys*, vol. 15, no. 3, pp. 237-270, Sept. 1983.
- [3] R. S. Michalski et al., Eds., *Machine Learning: An Artificial Intelligence Approach*. San Francisco: Tiogo, 1983.
- [4] G. E. Hinton, "Connectionist learning procedures," *Artificial Intell.*, vol. 40, pp. 185-234, 1989.
- [5] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, pp. 1134-1142, 1984.
- [6] R. S. Michalski, "Pattern recognition as rule-guided inductive inference," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-2, no. 4, pp. 349-361, July 1980.
- [7] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
- [8] K. S. Narendra and R. M. Wheeler, "An N-Player sequential stochastic game with identical pay-offs," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, pp. 1154-1158, Nov. 1983.
- [9] J. S. Quinlan, "The effect of noise on concept learning," in *Machine Learning: An Artificial Intelligence Approach*. (R. S. Michalski, et al., Eds.) San Francisco: Morgan Kaufman, 1986, pp. 149-166, ch. 6, vol. 2.
- [10] R. Rada and R. Forsyth, *Machine Learning: Applications to Expert Systems and Information Retrieval*. Chichester, UK: Ellis Horwood, 1986.
- [11] S. R. Ranjan, "Stochastic automata models for concept learning," M.S. Thesis, Dept. Elec. Engg., Indian Inst. Sci., Bangalore, India, Mar. 1989.

- [12] S. R. Ranjan and P. S. Sastry, "A parallel algorithm for concept learning," in *Frontiers in Parallel Computing*, V. P. Bhatkar et al., New Delhi: Narosa, 1991, pp. 349-358.
- [13] A. G. Barto, "Learning by statistical cooperation of self-interested Neuron-like computing elements," COINS Tech. Rept., Dept. of Comput. Inform. Sci., Univ. Massachusetts, Amherst, Apr. 1985.
- [14] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
- [15] M. A. L. Thathachar and P. S. Sastry, "Learning optimal discriminant functions through a co-operating team of automata," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, no. 1, pp. 73-85, Jan.-Feb. 1987.
- [16] K. Rajaraman, "Learning automata models for concept learning," M.E. Thesis, Dept. of Elect. Eng., Indian Inst. of Sci., Bangalore, India, Jan. 1991.
- [17] S. Mukhopadhyay, "Group behavior of learning automata," M.E. Thesis, Dept. of Elect. Eng., Indian Inst. of Sci., Bangalore, India, Jan. 1987.
- [18] D. Haussler, "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework," *Artificial Intell.*, vol. 36, pp. 177-221, 1988.

Multiple Participant-Multiple Criteria Decision Making

Keith W. Hipel, K. Jim Radford, and Liping Fang

Abstract—Meaningful connections among different types of decision making situations are established in order to improve the development of useful decision technologies for application to real world problems. More specifically, an assertion is put forward that suggests that single participant-multiple criteria (SPMC) and multiple participant-single criterion (MPSC) decision making problems may be treated in essentially the same way. In this way, decision technologies that are already available can be suitably refined for use in studying both SPMC and MPSC decision situations. Another assertion is made that a multiple participant-multiple criteria (MPMC) decision situation can be converted to a MPSC decision situation. In order to simplify the explanation, illustrative applications are utilized throughout the paper. Finally, worthwhile directions for future research are summarized in the last section.

I. TYPES OF DECISION SITUATIONS

In decision making, a perceived solution to a given problem is selected from a set of possible alternatives. Moreover, every decision situation exists in an environment. This environment consists of a set of circumstances and conditions that affect the manner in which the decision making problem can be resolved. There are four major factors that determine the characteristics of this environment, namely:

- 1) Whether or not uncertainty exists in the decision situation being considered,
- 2) Whether or not the benefits and costs resulting from the implementation of available courses of action can be completely assessed in quantitative terms,
- 3) If a single objective is involved or if multiple objectives must be taken into account,

Manuscript received November 6, 1992; revised January 4 1993. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

K.W. Hipel and J. Radford are with the Department of Systems Design Engineering, University of Waterloo, Waterloo, ON, Canada N2L 3G1.

L. Fang is with the Department of Systems Design Engineering, University of Waterloo, Waterloo, ON, Canada N2L 3G1 and also with the Department of Mechanical Engineering, Ryerson Engineering University, Toronto, ON M5B 2K3 Canada.

IEEE Log Number 9209640.

- 4) Whether the power to make the decision lies in the hands of one organization, individual or group or whether more than one of the participants have power to influence the outcome of the situation.

All possible combinations of these four factors lead to 16 categories of decisions. These 16 categories can be condensed into five, as shown in Table I [15, p. 7].

The first category shown in the left-hand column of Table I consists of decision situations in which there is, or there is assumed to be, no uncertainty. A single decision maker having a single objective must make a choice among alternatives on the basis of an evaluation of these alternatives in quantitative terms. Examples of this type of decision occur in many routine operational and administrative processes. A wide range of mathematical models, including linear, nonlinear and dynamic programming, are available to assist managers faced with these kinds of decisions.

In Category 2 in Table I, no quantitative measures of benefits and costs are available. In such circumstances, there is no well-defined and generally-accepted measure of the benefits for the alternative choices that could be made. Without such measures, no unequivocal basis of choice among the alternatives is available. For example, how can I choose between having five apples or four oranges when I am not sure what is the best way to compare them in order to make my selection?

In Category 3, estimates of probability are used to deal with uncertainty. The decision required is between say 1) a 30% chance of winning \$10 000 (with the corresponding 70% chance of winning nothing) and 2) a 60% chance of winning \$6000 (and a 40% chance of no profit at all). If the opportunity described occurs many times, the decision can be made on the basis of the expected benefits of the two alternatives. In these circumstances, the alternative with a 60% chance of obtaining a \$6000 profit is that which provides the greatest profit over many repetitions of the same situation. However, if the situation occurs only once, this basis for choice is not appropriate. The selection between alternatives in single-occurrence situations of this sort depends on the attitude of the decision maker toward risk, rather than on a comparison of expected values.

Decision making in Category 4 is even more complicated by virtue of the existence of multiple objectives or multiple criteria for decisions. In the example shown in Table I, the decision maker must choose either a new set of golf clubs, an ocean cruise or a new suit of clothes. He can afford only one of the alternatives. Three objectives are involved; a) to improve performance in a recreational activity; b) to improve health and well-being; and c) to present an improved image to the world. Neither the benefits of adopting any one course of action nor the chance that the benefits envisaged will be attained are necessarily expressible in quantitative terms. The ultimate choice in such situations is made by individuals using judgement and intuition with respect to the information available at the time of decision.

The decision situations in Category 5 constitute the most complex scenarios with which society is faced today. These situations involve two or more participants, each having his or her own objectives and intentions and each endeavouring to bring about his or her most preferred outcome. These situations are encountered in planning and policy making in corporations, in public policy development and in many other circumstances in modern-day life. They occur within organizations and communities and amongst them. Moreover, they are often inter-linked with other circumstances of the same sort, sometimes involving all or some of the same participants. The