Acquiring Rule Sets as a Product of Learning in a Logical Neural Architecture

Michael J. Healy, *Member, IEEE,* and Thomas P. Caudell, *Member, IEEE*

***Abstract*— Envisioning neural networks as systems that learn rules calls forth the verification issues already being studied in knowledge-based systems engineering, and complicates these with neural-network concepts such as nonlinear dynamics and distributed memories. We show that the issues can be clarified and the learned rules visualized symbolically by formalizing the semantics of rule-learning in the mathematical language of two-valued predicate logic. We further show that this can, at least in some cases, be done with a fairly simple logical model. We illustrate this with a combination of two example neural-network architectures, LAPART, designed to learn rules as logical inferences from binary data patterns, and the stack interval network, which converts real-valued data into binary patterns that preserve the semantics of the ordering of real values. We discuss the significance of the formal model in facilitating the analysis of the underlying logic of rule-learning and numerical data representation. We provide examples to illustrate the formal model, with the combined stack interval/LAPART networks extracting rules from numerical data.**

***Index Terms*— Adaptive resonance theory (ART) neural networks, classification, connectionist system, formal semantics, inferencing, knowledge-based systems, logic, predicate, rule base, rule extraction, synaptic learning.**

I. Introduction

WE PRESENT a formalization of the semantics of certain classes of neural networks in two-valued logic. The need for such a formalization arises from the architectural models proposed by some investigators [1]–[4] as systems which learn rules, in the sense of a knowledge-based system. Rule-learning by neural networks is an ambitious goal, for it requires an interpretation of adaptation in a connectionist system as the learning of inference relations expressing cause–effect, condition–action, or other antecedent-consequent relationships understandable in human terms. In this paper, we propose a formal but relatively simple model to support this interpretation for a neural network whose learned rules are readily analyzed in two-valued logic, yet can express complex phenomena represented by real-valued data. The formalization is intended as an analysis tool, to support the clarification and possible resolution of the issues we discuss.

We emphasize at the outset that this is a theoretical paper aimed at the analysis of practical applications. The aim here is not to present specific neural-network models or performance comparisons between them, but to present a formalization that helps clarify the issues in the symbolic representation of neural-network processing. Our method is to present a relatively simple, "user-friendly" formal model using as little mathematical notation as possible while correctly representing the semantics of the data and of the neural-network processing of the data. To that end, relatively simple neural-network architecures for data representation and rule extraction are used to exemplify the formalization.

Manuscript received March 19, 1996; revised September 15, 1996 and October 20, 1996.

M. J. Healy is with Research and Technology, The Boeing Company, Seattle, WA 98124-2207 USA.

T. P. Caudell is with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131 USA.

Publisher Item Identifier S 1045-9227(97)02744-6.

A. Rules and Neural Networks

The rule forms of knowledge-based systems and the strategies they employ for executing rules are many and varied [5], [6], but central to all rule-based systems is an *inference engine* that implements the system's rule-execution strategy. The most common form of rule is the general, antecedent-consequent form $A \rightarrow B$, suggestive of a deductive inference in formal logic (although the system itself is usually based upon heuristics). A set of rules for a given application forms a *rule base* which changes from application to application and can be incrementally updated as more is learned about an application. The inference engine determines which of possibly many rules apply when the user supplies an input query. It then decides which of these rules to process, or else prioritizes them for application one at a time. It processes the selected rules in either antecedent–consequent order (forward chaining), producing consequents as conclusions, or the reverse (backward chaining), finding antecedents that apply to a given conclusion. In forward-chaining, for example, a rule *succeeds* for input x if its antecedent A is a statement that applies to x. As a result, the system infers that B applies to x as well. Through a chain of such deductions, the user obtains an answer to the query in the form of a final conclusion.

An adaptive neural network learns such inference relations through *synaptic learning*, in which it modifies the weights of the synaptic connections between its nodes. The modifications are the result of the simultaneous activation of connected nodes. The activation is in turn the result of the network's processing of example application data cases in which the as-yet-unknown rules apply.

Unfortunately, the current state of neural-network technology does not include a generally accepted model describing the processing in a neural network in terms of unambiguous symbolic expressions for rule antecedents, consequents, and the $\rightarrow$ inference relation. When employing neural networks as knowledge-based systems, however, it is important to have such a model. For one thing, it is often desired to have an *explanation capability* for a rule-based system, allowing users to see the line of reasoning leading to a conclusion as well as

the conclusion itself. More generally, it is important to have a model supporting knowledge-based verification and validation [5]. Several issues arise in knowledge-based systems, concerning potential pitfalls such as inconsistent rule sets. For example, there might occur an instance x in which one rule concludes B and another rule concludes not B. The rules in a rule base are often related through logical relationships that can lead to such difficulties. The larger the rule base, the more difficult it is to discern troublesome relationships by simple means. Further discussion and examples of this are given in [5], [6]. Addressing these issues requires a formal model of the relationships among symbolic expressions which represent rules and their applications.

A formal model of rule-learning in neural networks adds a new dimension to the modeling task. A mathematically precise representation must be found not only for symbolic expressions, but also for neural-network quantities such as numerical connection weights and for the relationships between these two fundamentally different kinds of entities. In addition, the *learning* of rules by a network requires a formalization of the learning problem itself: How are symbolic logical relationships to be learned from data? How can this occur in a connectionist system?

B. A Formal Model

Formal models for neural networks have been proposed (see [7] for a rather sophisticated model based upon nonmonotonic logic and [8] for one that relates to fuzzy logic). Formalizations can require some time for familiarization, particularly when they involve mathematical models that are not in common use. We propose a relatively simple formalization in "traditional" mathematical logic and apply it to the investigation of rule-learning in neural networks. Because of space limitations, we present only the essentials needed to understand its application to our example architecture. Hence, we avoid some of the issues that can complicate logical models and caution that ours may require extension to encompass neural-network models that vary widely from that presented here. However, we have tried to be careful in our definitions and use of terms and hope this discussion helps clarify important aspects of rule-learning.

The LAPART (LAteral Priming Adaptive Resonance Theory) neural network [9] establishes relationships between labeled sets, or *classes*, of objects through synaptic learning. The fundamental connection between this and purely symbolic processing is that statement symbols such as A and B, introduced earlier, are labels for two equivalent conceptual entities: classes of objects and logical functions. There are various reasons for our adopting the term "class": noise and other phenomena in the application data can cause many patterns to represent one object, data limitations can cause one pattern to represent many objects, and it is often desirable to establish groupings of objects that are examples of a single important concept. The use of classes allows a two-level representation—object and class—for flexibility in dealing with these occurrences. The logical functions, called *predicates*, yield either of two Boolean values, true or false, for a given argument: That is, $A(x) =$ true if and only if an object denoted by x is a member of class A, and $A(x) =$ false otherwise. To formalize the concept represented by the argument x, we have introduced the term *objects* to refer informally to members of classes. Normally, objects are the events or items represented by neural-network input, output, and connection-weight patterns, including possibly the patterns themselves.

We can now reformulate the task of modeling the learning of rules of the form $A \rightarrow B$: We seek a symbolic expression for a statement $B(z)$ expressing membership in class B for some object z, and this expression is to be a logical consequence of another expression A(x) expressing membership of an object x in some class A (we shall simplify by eliminating the object symbol z in Section III, but prefer to retain it for now). The data for a given instance of the rule are given as a pair of input data patterns representing an instance (x, z) of the rule. We have created a neural network called LAPART, specifically designed for learning logical inferencing relationships from data according to this scheme. A LAPART network consists of two interconnected adaptive resonance theory version 1 (ART 1) networks [10]. An ART 1 network learns to recognize classes of inputs through unsupervised learning, in which it derives pattern features to represent object classes based upon examples whose patterns appear *similar* according to a criterion implemented by the network. A LAPART network learns logical inference relations between object classes by hypothesis-testing, in which it reads example pairs of data patterns and forms and executes trial rules (Fig. 1). Each ART 1 subnetwork reads a single pattern from each pair. It incorporates the pattern into a cluster, corresponding to an object class. The connections between the two ART 1 networks force the classes to reflect correct rules. That is, a class instance (object) tentatively recognized by the first network A triggers a rule inferring that an instance of an associated class (an as yet unseen object) must be recognized by the second network B. Operating as just described, a LAPART network performs 1) partially supervised ART 1 pattern classification, resulting in synaptic learning within each ART network and 2) synaptic learning of the class-to-class inferences, or rules, through interconnects between the ART 1 networks.

C. On the Contents of this Paper

The remainder of this paper focuses on a formal model of rule extraction with a LAPART network and an associated preprocessing network. The formalization is in the form of formulas in a two-valued logic. We have intentionally omitted discussion of probabilistic reasoning, fuzzy logic, and other models of uncertainty on the grounds that they involve very complex issues and would greatly increase the length of the paper. We also omit discussion of "certainty factors" for rules, an attempt at handling uncertainty in expert systems. For a discussion of the latter, including a more general discussion of uncertainty in models of probabilistic as well as heuristic inferencing, see [11]. Our formalization of the semantics of rule extraction with neural networks will be entirely in symbolic statements with a "true–false" interpretation, leaving open the issues of uncertainty. We also omit discussion of other architectures such as fuzzy ARTMAP [12] that are functionally similar to LAPART. Fuzzy ARTMAP, for example, learns maps between multidimensional spaces represented in real as

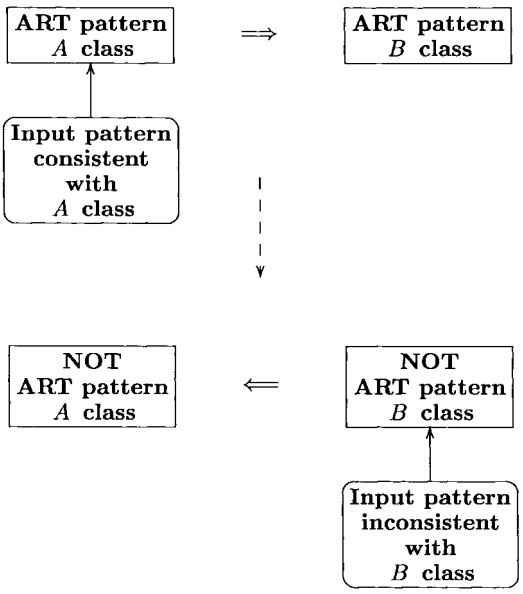


Fig. 1. The forward and backward inference rules implemented by a LAPART network.

well as binary patterns. A discussion of the semantics of these maps—similar to our rules—is beyond the scope of this paper, since *formalizing* these as learned rules is a more involved task than is that for LAPART.

In Section II, we present a representation of ART 1 classification in formal logic. The basic idea is that if learning has reached a *goal state* (i.e., the objects represented by the input patterns are completely represented by the learned templates), each template component represents a necessary condition for an object to be a member of a class and conjunctions of these conditions formalize the concept of generalization. We go on to describe symbolic rules as learned inferences between classes of objects in Section III. We define stack interval networks, which encode real-valued pattern components as binary subpatterns of a data pattern, in Section IV. As the formalism shows [13], [14], the significance of these networks is that they capture the semantics of the ordering of numbers, a necessary feature for data classification with real values. Section V contains simulation examples and illustrates rule formation with some visualization aids, and Section VI is the conclusion.

II. ART 1 and Logic

We shall assume a basic familiarity with ART 1; it is described in many places [9], [10], [15], and [16] significantly extends the results concerning its learning behavior. When acting as a stand-alone system, an ART 1 network autonomously clusters binary input patterns. In this section, we interpret binary patterns as descriptions of objects in predicate logic and then show how pattern clustering can be interpreted as the formation of object classes.

A. Logic and Neural Networks

Neural-network nodes that have an activation threshold can be regarded as logical functions of the neural events that impinge upon them (see [17] and [18]), and we argue that synapses can also. For a node, the events are connection-weighted input signals; for a synapse, the events are patterns of activity in the network that determine whether or not the activity of its source node determines the activity of its target node. Taking the view that neural events originate from things that are somehow encountered in an application, we allow the term "objects" to encompass all such events. A function evaluation yields a *proposition*, a statement with an assigned Boolean `true` or `false`, depending upon whether or not the current input object possesses the property represented by the function, or predicate. A network node can be intrepreted as a predicate by regarding its output activation as a binary ON/OFF signal, where ON (the value `true`) occurs when the numerical sum of inputs to the node exceeds a threshold value. The semantics of the predicate—what it means in the context of the neural network and its data—is determined by the input connection weights, or by the properties of a sensor if the node is an input node, and is also determined by the threshold value. Through their collective action, these quantities implement a decision process to determine whether the object represented by the current input to the node possesses some property—a particular color, a brightness level, or more generally, membership in some class uniquely determined by a set of properties.

This underlies the connection between Boolean values and the numerical values in data patterns, connection-weight patterns, and patterns of ON/OFF activation over the network nodes. A binary input pattern $\tilde{I}(x)$ input to an ART 1 neural network, representing an object x, can be regarded as a string of "1's" and "0's." Alternatively, it can be regarded as a list $I(x)$ of Boolean values `true` and `false`, respectively, representing truth or falsehood: Each input node "makes a statement" that x either does or does not appear to possess the unique property that it represents. Correspondingly, we use an identical notation to denote two different operations (it will always be clear which is intended): The numerical minimum and the logical AND, both denoted $\wedge$. For any two binary patterns $\tilde{X}$ and $\tilde{Y}$ having the same number of zero–one components, or *length*, let $\tilde{X} \wedge \tilde{Y}$ denote the binary pattern that constitutes their componentwise minimum, where $0 \wedge 0 = 0, 1 \wedge 1 = 1, 0 \wedge 1 = 0 = 1 \wedge 0$. For propositions p and q, on the other hand, let $p \wedge q$ denote their logical AND, or *conjunction*, which has the value `true` or `false` depending upon whether both statements are true or not.

Let $\tilde{I}(x)$ denote an arbitrary binary input pattern to an ART 1 network with n input nodes; there being one pattern component per input node, the pattern is said to have length n. The corresponding list of n Boolean values $I(x)$ indicates whether x possesses each in a list of elemental properties I_k (where $1 \leq k \leq n$). The values are derived via application data preprocessing, and can characterize any of a number of items—image brightness or color values, accelerometer signal amplitudes, disease symptoms or diagnostic test results, and so on. The pattern component $\tilde{I}_k(x)$ is a binary one or zero, indicating whether or not $I_k(x) = \texttt{true}$. Informally, $I_k(x)$ is the statement "x is observed to have property k."

Based upon $\tilde{I}(x)$ and the current state of its synaptic memory, the ART 1 network assigns x to a class of objects whose patterns are similar to a binary *template* pattern for

the class. Here, similarity means that $\tilde{I}(x)$ and the template share a sufficiently large subpattern of binary "1's." Each class template is a pattern of synaptic connection weights located in a distinct set of connections, separate from the connections for the other class templates. Being adaptive, the network will "learn" the information in $\tilde{I}(x)$ by updating the matching template's synaptic weight values, thereby affecting the network's future classification of objects. In this way, the network develops a classification scheme for objects from a sequence of input examples.

The actual selection of object properties to be represented by the predicates I_k is an important consideration. The descriptive value of the selected properties has a great impact upon the effectiveness of the network in identifying objects in the user's application, and achieving an appropriate selection may dictate considerable preprocessing of the available data. The logic of the network, on the other hand, is independent of any particular selection of properties. In analyzing this logic, it is enough to discuss the manner in which ART 1 interprets the patterns, with the properties regarded as abstract propositions.

B. The ART 1 Algorithm

The ART 1 classification algorithm finds an object class to represent its current input x based upon the binary input pattern $\tilde{I}(x)$ that represents x. The persistent activation of an F_2 node $F_{2,J}$ during the input of $\tilde{I}(x)$ signifies that x has been assigned a class. The algorithm simply describes a procedure for computing the class index J. Let $\|\tilde{X}\|$ denote the sum over the n zero–one components in a binary pattern $\tilde{X}, \|\tilde{X}\| = \Sigma_{i=1}^{n} \tilde{X}_i$. Let β be a small, positive value less than $1/n$, where n is the number of input nodes I_k. Let ρ be the ART 1 vigilance parameter, with $0 \leq \rho \leq 1$. For a given binary input pattern $\tilde{I}(x)$, the ART 1 algorithm specifies a class node $F_{2,J}$ and its associated binary template pattern $\tilde{T}^J$ by solving the combinatorial optimization problem

$$\begin{aligned} \underset{j}{\text{maximize}} \quad & \frac{\|\tilde{I}(x) \wedge \tilde{T}^j\|}{\beta + \|\tilde{T}^j\|} \\ \text{subject to} \quad & \frac{\|\tilde{I}(x) \wedge \tilde{T}^j\|}{\|\tilde{I}(x)\|} \geq \rho. \end{aligned} \tag{1}$$

The solution node $F_{2,J}$ serves as a label for a class of objects. Formally, its persistent activation assigns the value `true` to the logical function evaluation $F_{2,J}(x)$, which denotes the proposition, "x is a member of class J."

C. Formalizing

To see how the template pattern components appear in the formalization, suppose that resonance has not yet occurred in obtaining a solution to (1), but that the current F_2 "choice" $F_{2,J}$ (for some J) is being evaluated via the ART 1 pattern-matching operation at the F_1 layer. The binary pattern minimum $\tilde{I}(x) \wedge \tilde{T}^J$, with components $\tilde{I}_k(x) \wedge \tilde{T}_k^J$, occurs as the current F_1 activation pattern, with $\tilde{F}_{1,k}(x) = \tilde{I}_k(x) \wedge \tilde{T}_k^J$. For each k, then, the corresponding logical function evaluation $F_{1,k}(x)$ represents the *conjunction*, or *logical* AND, of $I_k(x)$ and a proposition corresponding to the template component $\tilde{T}_k^J$ (where J is the index of the tentatively selected F_2 node). We express the proposition corresponding to $\tilde{T}_k^J$ as follows:

$$(\forall y) F_{2,J}(y) \Rightarrow I_k(y). \tag{2}$$

The subexpression $(\forall y)$ is the familiar universal quantifier applied to the variable y, so that the implication formula $F_{2,J}(y) \Rightarrow I_k(y)$ reads, informally, "For all y, if y is a member of class $F_{2,J}$, then y has property I_k." The variable x in $I_k(x)$, on the other hand, represents only the *current* input object. The occurrence of the conjunction at $F_{1,k}$ is formalized as follows:

$$F_{1,k}(x) = (I_k(x) \wedge ((\forall y) F_{2,J}(y) \Rightarrow I_k(y))). \tag{3}$$

The $=$ sign here means that the left and right hand expressions have the same Boolean values `true` or `false`. The conjunction on the right-hand side in (3) has the value `true` if and only if $I_k(x) =$ `true` and $((\forall y) F_{2,J}(y) \Rightarrow I_k(y)) =$ `true`. Equivalently, from the correspondence between logical truth values and binary values, the binary activation value of node $F_{1,k}$ is one, $\tilde{F}_{1,k}(x) = 1$, if and only if $\tilde{I}_k(x) \wedge \tilde{T}_k^J = 1$.

D. Learning

The logical implication formulas (2) state putative necessary conditions for class membership. In general, an object can be adopted by a class (its binary pattern can resonate with the class template) even though several of these conditions are violated, hence, they do not characterize fully the ART 1 network's behavior—that is taken care of by the optimization formulation (1). What the template formulas do characterize is a hypothetical goal state of learning, in which all objects in an application will have been observed. As shown in [10], the small value for β used in (1) ensures that F_2 nodes whose templates are *subset* templates for the current input pattern will be preferred in the F_2 competition. Here, the term "subset" has the obvious meaning: A subset template has component value "1's" only where the input pattern has "1's." If the largest subset template for an input pattern does not activate the vigilance system, then it will occur immediately as the resonant template. This is called the *direct access property*. One way of explaining this property is to say that the ART 1 system has reached a goal state of learning in an application when the patterns representing the objects directly access their resonant templates.

During a resonance, the conjunctions (3) are synaptically learned by the network by adapting the template weight values to equal the current F_1 pattern. This is expressed in the following binary pattern equation:

$$\tilde{T}_{\text{new}}^J = \tilde{I}(x) \wedge \tilde{T}_{\text{old}}^J = \tilde{F}_1^{\text{resonance}}(x). \tag{4}$$

The modified template value $\tilde{T}_{\text{new},k}^J$ is the Boolean value of the kth conjunction in (3). When a class is first established, all connection-weight values in its template are "1's," that is, all the implications (2) are putative necessary conditions in the absence of any learning. Many of them are found to be `false` as the network learns from examples, and the corresponding template weights are therefore set to zero. We express this monotonicity in the synaptic learning of ART 1 by saying that an ART 1 network proceeds toward a goal state of learning by deleting necessary conditions that have been found not to apply to a class.

In our analysis, the putative necessary conditions expressed by the "1's" in a template are generalizations about the objects in a class. Let C_J be the set of indices corresponding to the surviving "1's" in a template $\tilde{T}^J$ at any given time, $C_J = \{k|\tilde{T}^J_k = 1\}$. Then a *conjunctive generalization* about the class J as of that time is the conjunction of putative necessary conditions

$$\bigwedge_{k\in C_J} ((\forall y)F_{2,J}(y) \Rightarrow I_k(y)). \tag{5}$$

This yields a formal definition of generalization for the ART 1 neural network: An ART 1 generalization about a class of objects is the conjunction formula whose index set C_J corresponds to the class template. If learning has reached a goal state, this template is the direct access template. That is, the index set for the conjunction has maximal size such that the conjunction is true for all objects in the class.

III. Learning the Rules

It can be seen from Fig. 2 that there are separate systems of lateral connections $A \rightarrow B$ and $B \rightarrow A$ between ART 1 networks A and B in a LAPART network. The roles of all these connections are described in detail in [9]; for brevity, we shall limit the discussion to two main sets: the $F_2^A \rightarrow F_2^B$ class-to-class connections, which are adaptive, and the $VIG_B \rightarrow VIG_A$ lateral reset connection, which is fixed. The other connections help implement the inference-learning (rule-learning) function of these main connections. The class-to-class connection weights store the learned inferences, and the lateral reset connection serves as the mechanism for supervised learning.

A. How LAPART Learns Inference Relationships

Suppose that the objects in an application are of two kinds, such as the weight of an adult bird and the average frequency with which the bird flaps its wings. Suppose further that if an object x_A of the first kind falls into some class A_i (where there are mutually exclusive classes $A_1, A_2, \cdots$), then it is always the case that some object x_B in a unique class B_j (where there are classes $B_1, B_2, \cdots$) occurs along with x_A. Suppose that a representation for individual objects of the two kinds is available in two sets of Boolean-valued properties. The objects can then be encoded as binary patterns, one for each object of the first kind using the first property set and one for each object of the second kind using the second property set. The motivation behind LAPART is that two ART 1 networks, properly interconnected and with the properties of each kind represented by the respective F_1 layers, might be able to identify the correct $A_i \longrightarrow B_j$ class-to-class inference relationships by processing binary pattern pairs corresponding to example pairs (x_A, x_B) of related objects. These inference relationships are the rules learned by the network.

To simplify the discussion, we shall assume that the two objects x_A and x_B in a related pair (x_A, x_B) are really the same object x. The A- and B-network F_1 properties will be regarded, then, as two sets of properties for a single set of objects. Let the predicate corresponding to a network A F_2 node $F^A_{2,i}$ be denoted A_i. Similarly, let the predicate corresponding to network B node $F^B_{2,j}$ be denoted B_j. Evaluating the F_1^A and F_1^B properties for an object x, we obtain a pair $(\tilde{I}_A(x), \tilde{I}_B(x))$ of binary input patterns for a LAPART network. Upon receiving its input pattern $\tilde{I}_A(x)$, network A performs the usual ART 1 hypothesis-testing, then resonates at an F_2^A node corresponding to some class A_i. If network A has correctly represented x, then, we can write $A_i(x) = \texttt{true}$. If that is the case, then the F_1^A layer will be allowed to persist with the resonating pattern of activation, $\tilde{I}(x) \wedge \tilde{T}^{A,i}$, and this pattern will be learned as the new version of template $\tilde{T}^{A,i}$ by network A. However, the supervision enforced by the lateral connections in LAPART may change this. There are two cases, depending upon whether the A_i-node has previously resonated with an input pattern.

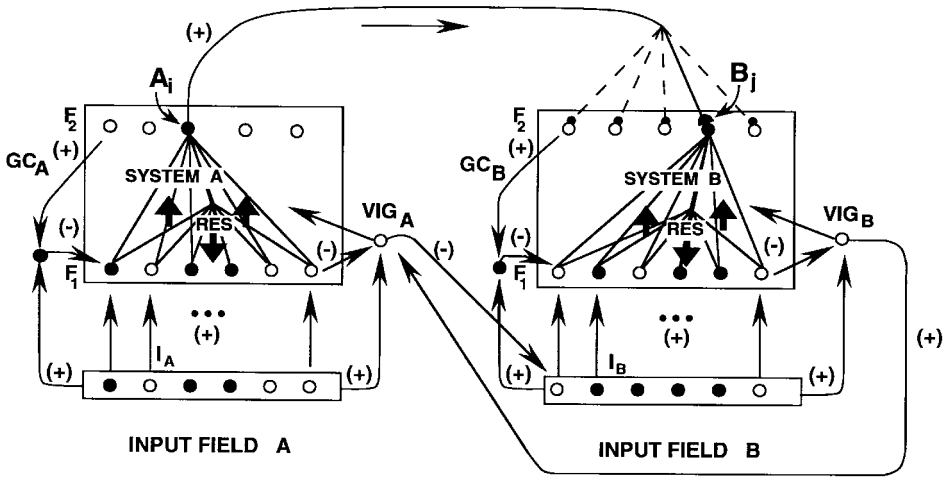
Consider first the case in which the A_i-node has not previously resonated with an input pattern. Then after a brief delay (network A has been allowed to suppress network B's input until it achieved resonance), network B will be allowed to receive its input pattern. Network B will then resonate on some F_2^B node representing a class B_j. The simultaneous resonance [Fig. 2(a)] leaves nodes A_i and B_j both active. As a consequence, synaptic learning proceeds quickly and the $A_i \rightarrow B_j$ ($F^A_{2,i} \rightarrow F^B_{2,j}$) connection strength approaches its upper bound value, which is a binary one. During any future presentation of an input pattern pair for an object z, if node A_i is the resonant node, it will send a strong signal to node B_j, selecting it as the exclusive class for z in network B. As a consequence, none of the $A_i \rightarrow B_k$ connections for nodes B_k with $k \neq j$ will have a chance to compete, and their strengths will remain at zero.

The second case occurs more frequently as the LAPART network learns rules: If node A_i has previously resonated, it possesses a learned connection to a single node B_j. Since network A delays network B's input, as mentioned before, there is a brief interval during which the hypothesis $A_i(x)$ and the always true statement $(\forall z)A_i(z) \Rightarrow B_j(z)$ are both called into play. This occurs through the persistent activation of the resonant node A_i in network A and its *lateral priming* of node B_j through the previously learned $A_i \rightarrow B_j$ connection. The consequent activation of node B_j is the inferred class membership statement $B_j(x)$. The laterally primed activation of B_j, with network B's input temporarily suppressed, forces network B to read out the template $T^{B,j}$ over the F_1^B layer [Fig. 2(b)], to be compared with its input pattern $\tilde{I}_B(x)$. System B still has control of its vigilance node, however, and can either confirm or disconfirm the choice made for it by network A.

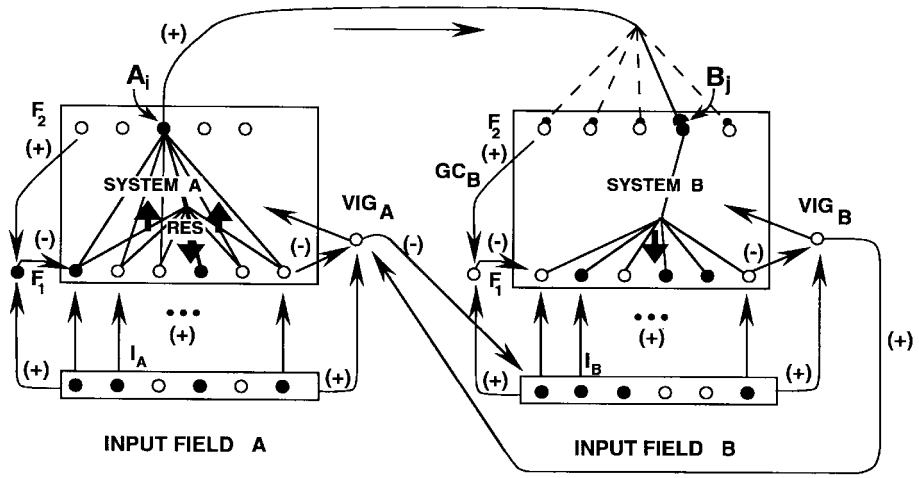
If the inferred class B_j is accepted by B, both networks, A and B, will update their templates in the ART 1 fashion

$$\tilde{T}^{A,i}_{\text{new}} = \tilde{I}_A(x) \wedge \tilde{T}^{A,i}_{\text{old}}$$
$$\tilde{T}^{B,j}_{\text{new}} = \tilde{I}_B(x) \wedge \tilde{T}^{B,j}_{\text{old}}.$$

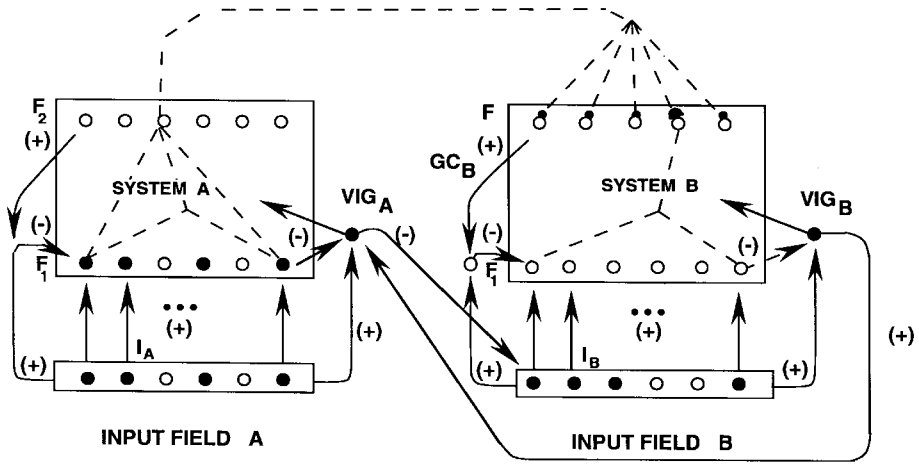
If the inference is not accepted, however, a lateral reset occurs: The vigilance system of network B, VIG_B, becomes active [Fig. 2(c); see also Fig. 1]. This causes a reset of network B, but also a reset of network A, because there is a direct, fixed



(a)



(b)



(c)

Fig. 2. Schematic of LAPART processing an input pattern pair. (a) New class A_i formed in network A; network B is allowed to act autonomously as an ART 1 system and select a resonating class B_j for its input; $A_i \rightarrow B_j$ inferencing connection set to maximum strength. (b) Previously formed class A_i already has a learned lateral connection to some class B_j. Resonance in network A causes B_j template to be read out over F_1^B via $A_i \rightarrow B_j$ activation of its F_2^B node. (c) If B input/B_j template is not a favorable match, VIG_B, acting through VIG_A as well as its connections to the F_2^B layer, mediates a lateral reset involving both the A and B networks.

connection from VIG_B to VIG_A. Network A is now forced to find a different class $A_{i'}$ (where $i' \neq i$) to represent the current input object x, and a new inference is made. Presentation of input pattern pairs as described thus forces the formation of pattern classes A_i in network A and B_j in network B in which class membership depends as much upon the learned

inferencing connections $A_i \rightarrow B_j$ as it does upon the ART 1 classification process.

The learning of a synaptic connection from the A_i node to a single F_2^B node conveys to LAPART the property that it generalizes from a single instance occurring in classes A_i and B_j to the rule

$$(\forall z)A_i(z) \Rightarrow B_j(z). \tag{6}$$

Because of the exclusive nature of B_j in network B, a complete formalization would include a formula stating that a representative for A_i in the B-network can exist *only* in class B_j. For brevity, we will instead make this implicit by the manner in which we apply the learned rules (6).

B. Applying LAPART to Rule-Learning

LAPART can be seen as a system for the autonomous capture and retrieval of knowledge in the form of rules that are implicit in masses of data. Let us refer to a LAPART network, together with its synaptic connection-weight memory at some stage of learning, as a LAPART system. A query to the system is a list of boolean or truth values $I_A(x)$ for an object x in the form of a binary pattern $\tilde{I}_A(x)$ input to subnetwork A. During rule learning and verification testing, $I_A(x)$ is accompanied by a list $I_B(x)$ input to network B, as a binary pattern $\tilde{I}_B(x)$. The ART 1 logic in network A is a forward-inferencing resolution strategy: the choice of an existing class A_i selects a rule to fire, in the form $(\forall z)A_i(z) \Rightarrow B_j(z)$ (if there is no appropriate class, a new rule is initiated). This results in the inference of an object class B_j. If $\tilde{I}_B(x)$ is present and the B_j template is not a good match for it, the ensuing lateral reset performs a backward inference via the contrapositive rule $(\forall z)\mathtt{not}B_j(z) \Rightarrow \mathtt{not}A_i(z)$ (see Fig. 1). By this process, LAPART creates, executes, tests, and modifies rules. By using either the forward inference as a query answer or the forward–backward inferencing as a rule-verification scheme (see [9]), LAPART also serves as an inference engine.

As a system for capturing and using knowledge, LAPART performs as a neural network in handling massive quantities of data, while the formal analysis shows that it clearly serves as a symbolic processor. LAPART's use as a forward-chaining inference engine is fairly straightforward: The fact that the learned A-classes are mutually exclusive provides an automatic resolution strategy for selecting a rule to fire for a given query—merely fire the rule $(\forall z)A_i(z) \Rightarrow B_j(z)$ if the input is a member of class A_i. However, the real value of LAPART is its role as an experimental neural architecture for rule-learning. In this role, it serves as a tool for the investigation of phenomena that can occur in systems that adaptively learn rules as logical inferences from data. Our formal model is being applied to this investigation.

C. Phenomena Encountered in Adaptive Inferencing

As mentioned in Section IIA, the data representation is a major factor in determining the applicability of the learning that takes place in an adaptive system. A simple neural network cannot be expected to perform all the tasks of data representation, classification, and class-to-class inferencing. In working with LAPART, we address this issue by using our knowledge of its adaptive inferencing algorithm as well as the application to guide the development of data preprocessing modules to achieve an effective representation. Our formalization shows [13] that the stack filter network works well with ART 1-based architectures in the representation of real and integer values in binary patterns.

Another important factor affecting applicability of the rules learned by a LAPART network is the order of presentation of examples. Different presentation orders can lead to different outcomes; this is an inherent truth in applying most classification algorithms, and the ART 1 algorithm is no exception. This effect can be undesirable in applications in which there is no intrinsic ordering of the data, such as for a static database. On the other hand, it can be advantageous if useable information is available on the effect that the order has upon classification: If this information implies an ordering that is better in some way than others, then this knowledge can guide the training strategy for the system. ART 1 provides some leverage here, for in [16] it is shown that learning in ART 1 follows a natural order related to input pattern size (number of "1's"). It would be interesting to see this kind of result incorporated in a formal model.

Another phenomenon that calls for a formal analysis is the inconsistency that can occur between the learned rules. As mentioned in Section I, inconsistency is a major issue in knowledge base verification. In LAPART, a learned inconsistency occurs when an input pattern $\tilde{I}_A$ is paired with two different patterns $\tilde{I}_B$ and $\tilde{I}'_B$, where the first of the latter two patterns does not trigger a lateral reset but the other one does. An occurrence of this sort could mean one of the following: 1) A mistake occurred in supplying input pattern pairs to the LAPART network. 2) The patterns were correctly generated, but the data are inconsistent due, for example, to noise. 3) There is no mistake and no inconsistency—it just happens that two different things can be inferred from the same input, given two different contexts. LAPART has no built-in recovery for occurrences 1) and 2). The use of a probabilistic mathematical model would help compensate for 1). A more complex network would be needed to deal directly with 2). This inconsistency must be detected by the network if it is to compensate for it, and the LAPART architecture does not currently provide for this: It only provides for detecting a classification error for a single object with a single class at a time. The cause of 3) is that the network A inputs do not contain enough information: An essential part of the context for an inference is missing. Analyzing an application to identify the missing context is a semantic problem, hence, is a task for formal semantics. The class templates and the lateral class-to-class inferencing connections formed by a LAPART network can be used to adavantage in such an analysis. Although it contains no mechanism for *signifying* an occurrence of inconsistency (signifying that a logical inconsistency had occurred would require a rather sophisticated architecture), LAPART does store the occurrence in the form of laterally connected templates.

Another phenomenon that can occur is as follows: Suppose that an A-input pattern $\tilde{I}_A(x)$ has a subset template for a class

A_i that would resonate with it, were it not for the presence of a larger subset template for class A_k, with $k \neq i$. Suppose further that the B_j template, which would have been selected by the rule $(\forall z)A_i(z) \Rightarrow B_j(z)$, would provide an acceptable template for the B-input pattern $\tilde{I}_B(x)$. However, it is the rule $(\forall z)A_k(z) \Rightarrow B_\ell(z)$ that is executed, and its inferred class B_ℓ template is a poor match, producing a lateral reset. Finally, suppose that at some time, it is desired to cease providing input patterns to network B and use LAPART purely as a forward-inference engine. Since no learning of any kind took place when the input patterns for both antecedent and consequent were available as just described, then the following can occur at some future time when the LAPART network is used purely in forward-inferencing. If the class A_i and A_k templates have not changed in their status as attractors for the pattern $\tilde{I}_A(x)$, and this pattern occurs as a query, then the incorrect inference B_ℓ will occur—no longer corrected for by the input of $\tilde{I}_B(x)$ to network B. This phenomenon can be detected by tracking individual pattern pairs through repeated cycles of presentation to the LAPART network and noting the repeated lateral resets that will occur with a pair that is subject to the capture phenomenon. If a lateral reset occurs every time a particular pair is presented, with the A-input pattern consistently causing the inappropriate rule involving a class A_k to fire as discussed, then the inappropriate subset template phenomenon has occurred. One can devise strategies for circumventing this phenomenon, but none of these is a substitute for a formal semantic analysis of the architecture *vis-a-vis* the data representation.

Our research goal is a better understanding of neural-network semantics through logical formalism. We hope that this discussion has helped in clarifying some of the issues involved and indicating how they might be elucidated by expressing neural computations as logical inferences.

IV. Encoding Analog Data-Stack Representation

Most applications of ART 1 networks require that data be preprocessed to obtain an effective representation of objects. In this section, we describe the stack interval network architecture, which converts integer- or real-valued data to binary input patterns. The binary patterns output by stack interval networks can be passed directly as input to one or both ART 1 networks in a LAPART network, enabling it to learn rules about objects represented by numerical data.

For properties which are already formulated as predicates, the preprocessing of data to obtain binary patterns is not difficult to conceptualize. However, application data consisting of real or integer values that are *meant* to be numerical values (as opposed to purely symbolic labels) must be preprocessed in a way that preserves the semantics of numbers: Integer and real numbers are totally ordered by the familiar $<$ (less than) relation. For any two numbers a and b, either $a<b, b<a$, or $a = b$. Then if numerical values are to be regarded as numbers rather than as abstract symbols, their neural-network representation must capture at least some essential property of the ordering relation. Whether the numbers are real or integer, it is reasonable to demand that the following property hold: If x, y, and z are such that $x<y$ and $y<z$, then the binary pattern for x must have more in common with the binary pattern for y than it does with the binary pattern for z. For an ART 1 network, this means that x and z can be in the same class only if y is also in that class.

Notice that no such relationship holds when the binary patterns are strings with the usual format used in digital computers, where the binary values represent the presence or absence of powers of two. For example, the numbers 122 and 128 represented as 8-b patterns in the powers-of-two format, with low-order binary digits to the right, are 01 111 010 and 10000000, respectively. The two numerical values are relatively close, yet a comparison of 1 b would show them to be relatively dissimilar—an ART 1 network, for example, would place them in separate classes regardless of its vigilance value. An example relating ART 1 pattern-matching to the semantics of the order relation is that the number 186, where $128<186$, has the powers-of-two representation 10111010, which matches the representation of 122 more closely than it matches the representation of 128.

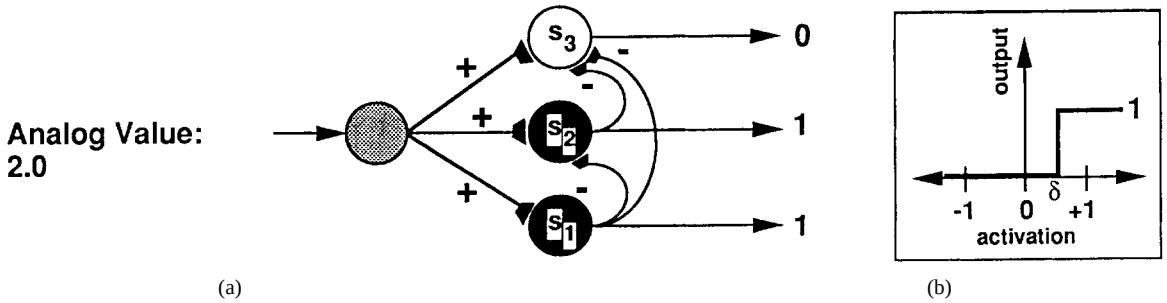
To properly encode numbers for ART 1 networks, we apply a basic representation scheme called a stack numeral. We then add more information, to achieve the stack interval representation. This is done as follows. First, a fixed lower bound ℓ and upper bound u are chosen for the data values x for the particular numerical property to be represented in binary patterns. Also, a positive integer d is chosen. Each data value x is transformed to a positive integer *stack value* m via the equation

$$m = \left\lceil \frac{x-\ell}{u-\ell} \cdot d \right\rceil \tag{7}$$

where $\lceil r \rceil$ forms the least integer greater than or equal to r. Next, we form a binary string of length d, with m "1's" beginning, say, at the left end of the string and with "0's" occupying the remaining positions to the right—hence the term "stack." For example, if we chose $\ell = 0, u = 128$ and $d = 128$, then 122 would have a stack representation as a string of 122 "1's" on the left and "0's" on the right, and 128 would be a stack of all "1's." Clearly, these two binary strings have relatively many binary values in common. Even more important is the fact that the binary positions which have value one in the stack representation of 122 are a subset of the binary positions with value one in the stack representation of 128. Stack numerals obey a "subset" relation that corresponds to the order relation $\leq$ for the numbers they represent. This property makes the stack numeral representation appropriate for the ART 1 algorithm (this is proved in [13]). In this connection, recall the direct access property of maximal subset templates.

Obviously, the stack representation gains in the effectiveness of representing numbers at the cost of larger binary patterns than the powers-of-two representation. The choice of the number d of stack levels is under the control of the user. However, high precision in representing numbers can require stacks—and, correspondingly, ART 1 networks—with many nodes and connections.

One step remains to compute a stack interval pattern. This step is to simply form the bit-by-bit complement of a



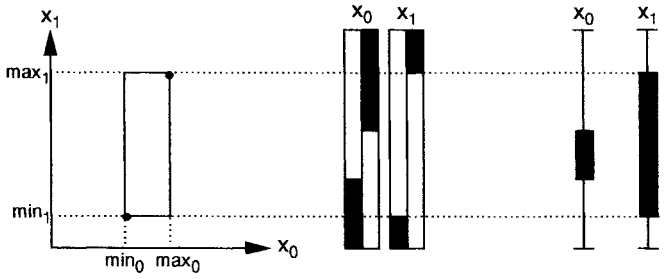
(a) (b)

Fig. 3. Schematic of a stack numeral network, which forms the positive stack in a stack interval network. (a) The inhibitory connections implement a binary pattern "subset" relationship that corresponds to the order relationship for subsets of the real number system. (b) The activation function of a stack node. $\delta = 0$ is used in the work reported here.

stack numeral and concatenate the two, obtaining a binary string of length $2d$. The reason for this will be explained presently. Using a simple example to illustrate, suppose that it was decided to represent numbers in the range 0–128 using stack intervals with $d = 10$. Then the number 115 would have the stack numeral representation 1 111 111 110. The complementary pattern is 0000000001, and the resulting stack interval pattern is 11 111 111 100 000 000 001.

A neural network that implements the stack interval representation (see Fig. 3) is described in [14], where it is proved that the combined stack/ART 1 neural-network architecture has the properties of Fuzzy ART. In the stack/ART 1 network, several stack interval networks convert numerical pattern values to binary stack interval patterns. Acting in parallel through one-to-one connections to the input nodes of an ART 1 network, the stack interval networks generate a composite binary input pattern for it. To within the discretization specified by the stack parameters ℓ, u, and d, the resulting templates correspond to the hyperbox regions formed from the numerical values by a fuzzy ART system operating in fast learning mode. A stack interval network representing a single numerical value that varies between the bounds ℓ and u with resolution d has $2d$ nodes and a system of fixed connections, inhibitory and excitatory. For brevity, we will describe only the connections for the stack numeral part of the network—a full description of the stack interval network is in [14]. The correspondence between stack interval templates and nonbinary hyperbox regions is shown in Fig. 4.

In a single stack interval network, the first d of the stack interval nodes represent the stack numeral half of a stack interval pattern. These are called positive stack nodes. The other d nodes, called negative stack nodes, represent the complement pattern. An item of numerical data can be thought of as a signal unit (a pixel in an imaging sensor that detects varying brightness levels, for example) whose output is a numerical value x. This unit has excitatory input connections, one-to-one and with unit strength, to each positive stack node in a single stack numeral network. Thus, each positive stack node receives x as input. The positive stack array represents the discretized stack value s as follows. To simplify the discussion, suppose that each node has activation threshold value $\ell = 0$ and that $u = d$ in the formula for converting x to s. Stack node 0, the lowest-order node, has inhibitory connections through which it provides, when activated, a unit



(a) (b) (c)

Fig. 4. Three visualizations of ART1 templates when stack interval networks supply the input representation: (a) hyperbox region representing a stack/ART 1 class in nonbinary (x_1, x_2) space, (b) positive and negative stacks forming a binary stack interval template pattern, from which the hyperbox is constructed, (c) range bar graph, constructed from each stack interval by simply drawing a vertical bar whose extent is equal to the "gap region" between the top of the positive stack and the bottom of the negative stack.

of inhibitory input to each of the other positive nodes. Node 1 has inhibitory connections to nodes 2 through d, node 2 has inhibitory connections to nodes 3 through d, and so on so that node m can receive as many as m units of inhibition from nodes 0 through $m-1$. Then, in the usual neural-network model in which a node becomes activated if the sum of its inputs—excitatory $(+)$ and inhibitory $(-)$—exceeds its threshold, each node becomes activated if and only if its input sum exceeds zero. Thus, stack numeral node m becomes activated only if node 0 becomes activated $(x>0)$. But because of the inhibitory input from node 0, node m then requires that $x>1$. But then, node 1 can overcome the unit of inhibition it receives from node 0, hence, sends an additional unit of inhibition to m. Proceeding through the stack nodes preceding m, we see that m requires $x>m$ in order to become activated, and, in that case, nodes 0 through $m-1$ will also have been activated. Notice that if $x \leq m+1$, nodes $m+1$ through d will remain inactive. Thus, nodes 0 through m will produce an output of magnitude one and $m+1$ through d an output of magnitude zero. If $x>d$, the entire positive stack will be activated, producing a row of "1's." This illustrates the operation of the stack numeral network in representing numbers in a specified range. The negative stack nodes implement the complementary representation through further use of inhibitory connections.

Stack interval patterns that represent numerical patterns are formed by concatenating the stack interval patterns represent-

ing the individual numerical pattern values. These composite patterns, and the ART 1 templates that form from them, can be visualized in four useful ways. Three of these are illustrated for a binary template pattern derived from binary input patterns representing two nonbinary values in Fig. 4. One way [Fig. 4(a)] corresponds to the "hyperbox" representation for fuzzy ART numerical pattern classes. The second [Fig. 4(b)] is a simple lineup of the alternating positive and negative stacks representing the numerical values. The third [Fig. 4(c)] combines the positive and negative portions to create a "range bar graph" on a normalized min/max linear graph. The ART 1 templates that form directly reveal the variation in numerical values in the input patterns that have contributed to their formation through binary pattern ANDing. For each numerical pattern component, the positive stack will have "eroded" to represent the minimum of all the numerical values in the patterns that contributed to it, and the negative stack will have "eroded" so that its bitwise complement represents the upper bound. Thus, each binary stack interval template pattern contains the information describing the total variation in input pattern components that contributed to its formation. That is, it represents an interval of nonbinary numerical values x_k that have contributed to the kth stack interval in the template (in Fig. 4, k takes the values one and two, since there are two nonbinary components).

This brings us to the fourth visualization, which is given directly in terms of the formal model of this paper: Each positive stack node represents a predicate of the form $s_{k,m} < x_k$ for some k, where $s_{k,m}$ is the nonbinary value represented by node m of stack k; this is the value x_k that exactly corresponds to m using the formula (7), where $0 \leq m < d$. The corresponding negative stack node represents the predicate $x_k \leq s_{k,m}$. Suppose that the positive and negative stacks corresponding to nonbinary component k have m' and $d-m''$ "1's," respectively, where $m' < m''$. Taking the complement of the negative stack, we see that the total stack interval pattern represents the interval $s_{k,m'} < x_k \leq s_{k,m''}$. Examples are given in Section V.

Although the discretization of real into binary values implemented by our stack network architecture is not a new idea (see, e.g., [1]), our formalization of it helps clarify its significance. Of further significance is the fact [14] that an ART 1 network with stack interval networks preprocessing analog pattern values for it (stack/ART 1) generates learned pattern classes equivalent to those of fuzzy ART [19] with fast learning, to within the resolution of the discretization of real values by the stacks. Given the practical limits to resolution of data values, this is an indication that fast-learning fuzzy ART classes are well-described in "ordinary" two-valued logic. The stack/ART 1 subnetworks of a stack/LAPART network form classes similar (although not identical) to those of the ART subnetworks of a fuzzy ARTMAP network.

V. Three Exaples of Rule-Learning

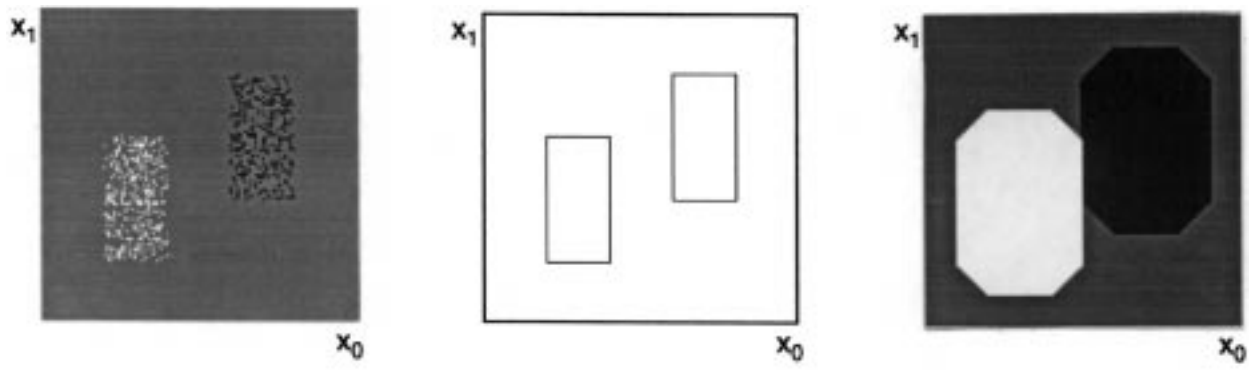
Three examples of rule-learning, or rule extraction, with the combined stack interval/LAPART architecture are given in this section. This is not meant as an indication of the performance of this example architecture *vis-a-vis* other architectures. It simply illustrates that our example architecture for formal analysis can perform in nontrivial applications. In fact, it will be seen that there are practical cases in which it performs nearly as well as optimal classifiers.

When one or both ART 1 subnetworks of a LAPART network are combined with stack interval networks, we call the composite network stack/LAPART. We generally use stack intervals to encode numerical data, and sometimes combine this with binary codes representing purely symbolic labels for properties of objects as well. A stack/LAPART network can learn rules which, because of the formal model presented in Sections II and III, can be easily extracted from the learned neural memories contained in the templates. Using the formal model for stack interval preprocessing networks in Section IV, the rules can include nonbinary numerical properties. In this section, we present three examples to illustrate this, all using stack interval patterns. Examples 1 and 2 are pedagogical in nature, and illustrate the visualization of templates as hyperboxes and range bar graphs. Example 3 is from an actual application, and illustrates rule extraction from complex data.

Example 1) Two-Dimensional Input Space, Two Disjoint Rectangular Distributions

The numerical patterns for network A have two components denoted x_1 and x_2. The numerical patterns for network B have a single component x_3. The B patterns might as well have been binary, since they represent purely symbolic labels for the network A classes and the labels are input directly through the input pattern pairs. Regarding the nonbinary numerical patterns themselves as objects to avoid further introduction of notation, let x be the object represented by a particular pattern pair. We denote the composite stack interval binary pattern input to network A by $\tilde{I}_A(x)$, representing x_1 and x_2, and the stack interval pattern input to network B by $\tilde{I}_B(x)$. The network B pattern represents a numerical value x_3 which is either high or low, depending upon which of two classes are prescribed for the corresponding network A pattern.

The points in Fig. 5(a) were generated from two disjoint, but otherwise random, distributions of (x_1, x_2) points in a two-dimensional region of Euclidean space. Each of the 800 points is contained in one of two rectangular regions, with corners at (0.2, 0.2) and (0.4, 0.6) for network A Class 1 and (0.6, 0.4) and (0.8, 0.8) for network A Class 2. Regarded as numerical patterns with two components, the pairs were coded as pairs of stack intervals having $d = 128$ positive stack nodes, resulting in 256 stack nodes total including the complement, for a total ART 1 input binary pattern size of 512. Each stack was normalized to represent the nonbinary interval $0 < x_k \leq 1$ for each dimension k of the data. These binary patterns were presented in random order to an ART 1 network with a vigilance value of $\rho_A = 0.65$. The network learned two templates (rules), plotted in Fig. 5(b) as hyperbox regions. Here, a single ART 1 network was used for network A and network B is not present, since the binary input patterns for the two mutually disjoint distributions are easily clstered into two classes by ART 1—LAPART is not needed. Fig. 6



(a) (b) (c)

Fig. 5. Two disjoint distributions in two-dimensional real space: (a) a scatter plot of the data samples (white for Class 1 and black for Class 2), (b) the learned class templates as hyperboxes, defined by their lower left and upper right corners, (c) rule attractor regions based upon the two hyperboxes (calculated using the ART 1 algorithm and stack parameter values).

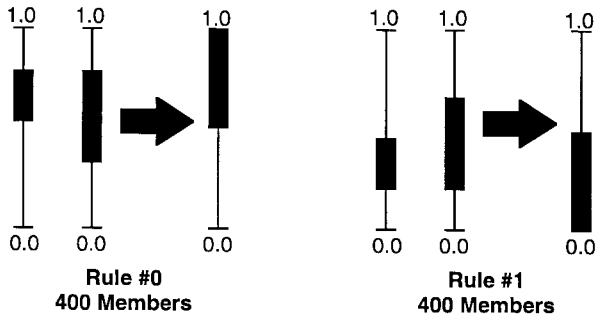


Fig. 6. Range graph representation of the two rules learned in Example 1.

shows a visualization of the two rules as range bar graphs. In this case, the range bar for each rule consequent is artificial; for Rule 0 (Class 1) it is "high" and for Rule 1 (Class 2) it is "low."

In addition to the hyperbox versus range bar visualization, this example aids in visualizing the connection between resonance, template modification, and the region of attraction of a hyperbox outside its borders (see also [20]). The region of attraction of points outside the boundary of a hyperbox is important when a goal state of learning has not yet been reached for a given collection of data points. Template modification comes about because templates are strong attractors for many binary patterns for which they are not subsets. Fig. 5(c) shows the regions of attraction for the two hyperboxes representing a goal state for the samples of the two distributions shown in Fig. 5(a). Following the reaching of this goal state, another set of points might be presented to the network, with many of the points lying outside the boundaries of the original two distributions. As long as the value of ρ_A does not change, many of the binary patterns representing these points will be attracted by one or the other of the two templates, and the attractor regions in Fig. 5(c) illustrate this (of course, the templates will also be appropriately modified, i.e., the hyperboxes will widen). Each attractor region for an A-template corresponds to the rule for which the template represents the antecedent, hence, we call each attractor region a "rule attractor." Note that a rule attractor has a distinctly larger volume than would be found for the corresponding rule if the rule were included in the rule base of a conventional forward-chaining rule-based system. A conventional system would require a point to be actually within a hyperbox for the rule to succeed.

Example 2) Two Overlapping, Normally Distributed Classes

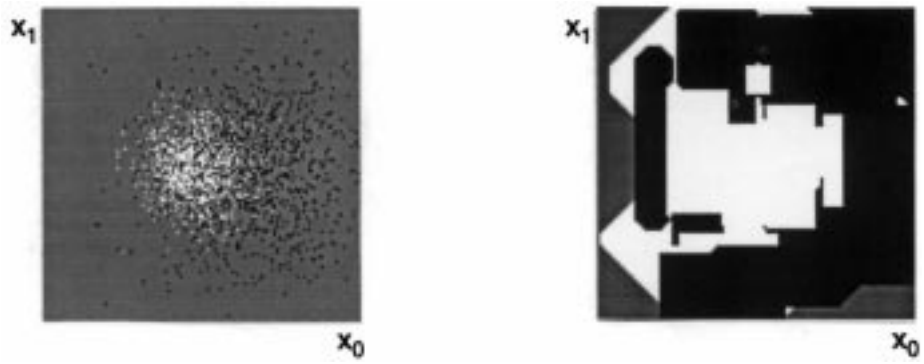
This example [21] illustrates the robustness of LAPART in applications with noise and/or confounding observations. Samples in a two-dimensional problem are distributed according to two overlapping multivariate normal distributions. The distribution parameters are the following.

Class 1: Mean $\mu_1 = (0,0)$, variance $\sigma_1^2 = 1$.

Class 2: Mean $\mu_2 = (2,0)$, variance $\sigma_2^2 = 4$.

Fig. 7 displays the randomly generated samples from the two classes. The Bayesian classifier optimal decision surface for this problem is a circle roughly centered on the narrower Class 1 and has an optimal probability of correct classification of approximately 81.5%. After the training of LAPART on 2000 samples total (1000 from each class), it was tested on 32 000 random samples equally divided among the two classes. To compare the performance potential of LAPART with the Bayes optimum, we investigated different values of ρ_A with this data. Since the distributions overlap, LAPART is needed here, but network B is simply presented with labels for the points—"high" for Class 1, "low" for Class 2—and ρ_B is set to one. For a fixed random ordering of the training data, and using a grid search with resolution of 0.01 for the network A vigilance value, the highest probability of correct classification was found to be 77.6%. This is less than four percentage points from optimal (the corresponding vigilance value was $\rho_A = 0.53$). This example illustrates that, even with the limitations discussed in Section III.C, LAPART can perform well as a classification system when noise and apparent contradictions are found in the training data.

The two examples just given illustrate the simple case of class labeling, which is the meaning often given to the term "classification." In this case, the rule antecedents are learned object classes but the consequents are simply labels. This does not capture the full generality of rule extraction, in which the



(a) (b)

Fig. 7. Two overlapping distributions in two-dimensional real space: (a) a scatter plot of the data samples, drawn from two normal distributions, (b) the rule attractors based upon the hyperbox regions for the two class templates. The optimal Bayesian decision surface is a circle surrounding the distribution with the smaller variance (white).

TABLE I

SUMMARY OF LAPART PERFORMANCE TESTS. PARAMETERS: RH = RELATIVE HUMIDITY; LC = LOW CLOUD AMOUNT; MC = MIDDLE CLOUD AMOUNT; HC = HIGH CLOUD AMOUNT; % ERROR = PERCENT MISSCLASSIFIED OR CLASSIFICATION NOT DETERMINED BY LAPART. TWO KINDS OF INFORMATION ARE SHOWN FOR EACH OF SIX DIFFERENT EXPERIMENTS: (1) THE X'S IN EACH COLUMN INDICATE WHETHER OR NOT THE PARAMETER WAS USED IN THE INPUTS TO LAPART AS A RULE ANTECEDENT IN THE EXPERIMENT DESCRIBED BY EACH ROW. (2) THE TOTAL ERROR INCLUDES BOTH MISCLASSIFICATIONS AND FAILURES TO FIND AN APPLICABLE RULE

t-3				t-2				t				t+3
RH	LC	MC	HC	RH	LC	MC	HC	RH	LC	MC	HC	%error
x	x	x	x	x	x	x	x	x	x	x	x	0.03
x	x	x		x	x	x		x	x	x	x	0.05
x	x			x	x	x		x	x	x	x	0.11
x	x			x	x			x	x	x		0.06
	x			x	x			x	x	x		0.09
	x				x			x	x	x		0.08

consequents are also learned object classes. The last example illustrates full rule extraction, with LAPART learning class-to-class inferences while simultaneously learning the classes.

Example 3) Weather Parameter Prediction

The example is taken from an actual application of LAPART to weather prediction. In the work of Soliz and Caudell [22], [23], a large data set consisting of nearly 1000 weather observations was used in the training. For this application, concatenated stack intervals were used to represent all non-binary values. The Cold Regions Research and Engineering Laboratory (CRREL), Hanover, NH, made available several series of weather data sets taken during events at Grayling, MI and Yuma, AZ. These data have been used to train the neural network and demonstrate the feasibility of a LAPART-based weather forecast model. For each of several data cases, a special subset of the data was used to train LAPART. A subset of weather state parameters was selected as antecedent (network A) parameters for each case. Observations of these parameters over a 6-h period were used to forecast a single parameter at a specified future time. The antecedent data were drawn from three time periods: the current time t in hours, and two times $t-3$ and $t-6$. Slightly more than 950 observations were processed by LAPART and associated with a future state parameter at $t+3$. Antecedent parameters were selected from a set that includes the amount of cloud cover for three cloud types and the relative humidity. The consequent (predicted) parameter was always the temperature.

To test the performance of LAPART in each case, nearly 5% (50 patterns) of the data were randomly selected and withheld from the training session. The withheld patterns were subsequently presented to the trained LAPART network and their mapping to the forecast parameter checked against the truth. The data for each case were reused in 20 tests, with a different randomly drawn subset of 50 patterns withheld for each test. The vigilance values were kept constant at $\rho_A = 0.75$ and $\rho_B = 0.80$, respectively. These values were found to yield rules each representing tens to hundreds of patterns, which was desirable. All variables were represented by stack intervals with $d = 16$ (see Section IV).

Table I summarizes the results of six different tests, each using the indicated parameters in the rule antecedents (e.g., LC, MC where "low cloud" and "medium cloud" parameters were used. The last column (labeled "% error") shows the percent of cases in which a test antecedent either yielded an incorrect prediction or generated a new class in network A (hence, no prediction) because it lay outside all of the rule attractors for templates (hyperboxes) that had formed during training.

Finally, representative rules are shown in logical form in Fig. 8, illustrating the nature of the hyperbox regions corresponding to the templates generated by LAPART. For complex problems with a high dimensionality and massive data bases, such as weather forecasting, the easily extracted symbolic form of the rules provides knowledge that is easy to interpret and to understand in human terms. Knowledge of

```
Rule 0
IF
    [ 0.00 < LC(t-6)  <= 0.50 ]  AND
    [ 0.00 < MC(t-6)  <= 1.00 ]  AND
    [ 0.00 < LC(t-3)  <= 0.90 ]  AND
    [ 0.00 < MC(t-3)  <= 1.00 ]  AND
    [ 68.0 < RH(t)    <= 92.0 ]  AND
    [ 0.00 < LC(t)    <= 1.00 ]  AND
    [ 0.00 < MC(t)    <= 1.00 ]  AND
             HC(t)    <= 0.00 ]
THEN
    [ -8.0 < T(t+3)   <= 48.0 ]

Rule 1
IF
    [ 0.00 < LC(t-6)  <= 0.80 ]  AND
    [ 0.00 < MC(t-6)  <= 0.70 ]  AND
    [ 0.00 < LC(t-3)  <= 0.70 ]  AND
    [ 0.00 < MC(t-3)  <= 0.65 ]  AND
    [ 32.0 < RH(t)    <= 80.0 ]  AND
    [ 0.00 < LC(t)    <= 0.40 ]  AND
    [ 0.00 < MC(t)    <= 0.70 ]  AND
    [ 0.00 < HC(t)    <= 0.40 ]
THEN
    [ 50.0 < T(t+3)   <= 82.0 ]
```

Fig. 8. Two of the rules extracted from the LAPART templates for the weather parameter prediction problem [23]. The human-readable logical form of these rules is easy to extract from the stack interval templates and the $A \rightarrow B$ inferencing connections learned from the data.

this kind is difficult to extract from experts using classical knowledge acquisition techniques.

VI. Conclusion

We have introduced a formal model for the underlying logic of symbolic processing with a class of neural-network architectures. We have show that this model characterizes key aspects of rule-learning from data with an example neural network called LAPART. Further, the model extends to patterns whose components are real-valued by combining LAPART with stack interval networks, whose formalization captures the order semantics of numbers. We have illustrated nontrivial rule-learning with results on data cases, including the learning of simple rules for weather prediction from meteorological data.

Our contribution is a formal model for symbolic processing from numerical data using neural networks. It is intended as an aid in the analysis of neural-network models, including the analysis supporting formal verification as applied to neural rule-based systems. The model requires careful extension to address these issues in other neural-network architectures. Extension to the formalization of uncertainty is also desirable. The relationship of our Boolean predicate logic model to fuzzy logic models of neural processing is not clear, but we have established a relationship between it and Fuzzy ART for quantized numerical values. The major advantage of boolean predicate logic is that it is well studied in mathematical logic and lends itself well to formal semantic models. We claim that the formal model is a potentially useful analysis tool for LAPART and similar neural-network architectures.

Acknowledgment

The authors would like to thank V. Lane for her help in preparing figures for this manuscript.

References

[1] S. I. Gallant, "Connectionist expert systems," *Commun. ACM*, vol. 31, pp. 152–169, 1988.
[2] S. Mitra and S. K. Pal, "Fuzzy multilayer perceptron, inferencing and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, 1995.
[3] J. Sima, "Neural expert systems," *Neural Networks*, vol. 8, pp. 261–271, 1995.
[4] G. A. Carpenter and A-H. Tan, "Rule extraction: From neural architecture to symbolic representation," *Connection Sci.*, vol. 7, pp. 3–27, 1995.
[5] K. L. Bellman, "The modeling issues inherent in testing and evaluating knowledge-based systems," *Expert Syst. with Applicat.*, special issue on verification and validation of knowledge-based systems, vol. 1, pp. 199–215, 1990.
[6] T. A. Nguyen, W. A. Perkins, T. J. Laffey, and D. Pecora, "Knowledge Base verification," *AI Mag.*, summer issue, pp. 69–75, 1987.
[7] G. Pinkas, "Reasoning, nonmonotonicity, and learning in connectionist networks that capture propositional knowledge," *Artificial Intell.*, vol. 77, pp. 203–247, 1995.
[8] R. Sun, "A neural-network model of causality," *IEEE Trans. Neural Networks*, vol. 5, pp. 604–611, 1994.
[9] M. J. Healy, T. P. Caudell, and S. D. G. Smith, "A neural architecture for pattern sequence verification through inferencing," *IEEE Trans. Neural Networks*, vol. 4, pp. 9–20, 1993.
[10] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, Image Processing*, vol. 37, pp. 54–115, 1987.
[11] J. Pearl, *Probabilistic Inference in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
[12] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural-network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Networks*, vol. 3, pp. 698–713, 1992.
[13] M. J. Healy, "On the semantics of neural networks," in *Adaptive Neural Systems: The 1992 IR&D Technical Report*, T. P. Caudell, Ed., Boeing Co., Seattle, WA, 98124-2207, Tech. Rep. BCS-CS-ACS-93-008, 1993.
[14] M. J. Healy and T. P. Caudell, "Discrete stack interval representations and fuzzy ART1," in *Proc. World Congr. Neural Networks*. Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. II-82–II-91.
[15] B. Moore, "ART 1 and pattern clustering," in *Proc. 1988 Connectionist Summer School*, D. S. Toretzky and G. E. Hinton, Eds. San Mateo, CA: Morgan Kaufman, 1989.
[16] M. Georgiopoulos, G. L. Heileman, and J. Huang, "Properties of learning related to pattern diversity in ART1," *Neural Networks*, vol. 4, pp. 751–757, 1991.
[17] W. S. McCullough and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.
[18] M. A. Arbib, *Brains, Machines, and Mathematics.*. New York: Springer-Verlag, 1987.
[19] G. A. Carpenter, S. Grossberg, and D. B. Rosen. "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.
[20] T. P. Caudell and M. J. Healy, "Studies of inference rule creation using LAPART," in *Proc. 5th IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, New Orleans, LA, 1996, pp. 1–6.
[21] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillian, 1994, pp. 165–176.
[22] P. Soliz and T. P. Caudell, "Application of artificial neural networks to battlefield atmospherics," in *Proc. 1995 Battelfield Atmospherics Conf.*, White Sands Missile Range, NM, Dec. 1995.
[23] ——, "Inferring future states of the atmosphere with a laterally primed adaptive resonance theory neural network," to appear in *Proc. 1996 INNS World Congr. Neural Networks*, San Diego, CA, Sept. 1966.

Michael J. Healy (M'93) received the M.S. degree in mathematics from the University of Idaho, Moscow, in 1967.

In 1967, he joined The Boeing Company, Seattle, WA, where he is now a Senior Principal Engineer. During his first 21 years with Boeing, he was a consultant in operations research and numerical analysis, specializing in numerical optimization. He worked on projects in preliminary design, computational geometry, economics, resource allocation, optimal flight control, and structural design. He also served as focal point for nonlinear programming for Boeing's Mathematical/Statistical Libraries. Since 1988, he has performed research and development in neural networks and formal methods applied to software engineering and knowledge-based systems. He started Boeing's ongoing research and development project in neural networks in 1989, and in 1994 initiated a research and development project in formal methods, applying category-theoretic methods to specification analysis and program synthesis. His neural-network research involves collaboration with the University of Washington, Seattle, Department of Electrical Engineering and the University of New Mexico, Albuquerque, Department of Electrical and Computer Engineering. He was appointed to the position of Affiliate Associate Professor in the University of Washington Department of Electrical Engineering in 1995. His research interests are in neural networks, learning theory, and applications of categorical semantics in computer science.

Mr. Healy is a Member of the American Mathematical Society (AMS), the Society for Industrial and Applied Mathematics (SIAM), and the International Neural Network Society (INNS). As a SIAM Member, he has served on the Advisory Board of the Optimization Special Interest Activity Group (SIAG/OPT), and he organized and chaired the first Minisymposium on Optimization and Supercomputing at the SIAM 35th Anniversary Meeting in 1987. More recently, he has organized and chaired sessions in Neural Networks at Northcon '90 and at IEEE Conferences on Neural Networks (ICNN '94) at Orlando, Florida and ICNN '96 in Washington, DC). He is an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS.

Thomas P. Caudell (M'91) received the bachelor's degree in physics and mathematics from California State University at Pomona in 1973, and the master's and Ph.D. degrees in physics from the University of Arizona, Tucson, in 1978 and 1980, respectively.

From 1981 through 1984, he was a Research Assistant Professor in the Department of Physics and Arizona Research Laboratories at the University of Arizona, Tucson, working in the field of solar seismology. From 1984 through 1989, he was a Senior Staff Physicist at the Hughes Artificial Intelligence Center, Hughes Research Laboratories, in Malibu, CA. There, he led Hughes' neural networks project and started their first virtual reality project. From 1989 through 1993, he was a Senior Principal Scientist in Research and Technology at The Boeing company, where he served as Principal Investigator on the Adaptive Neural Systems IR&D and associated technology transfer projects and performed research in virtual reality. He served as an Affiliate Associate Professor in the Department of Electrical Engineering at the University of Washington during that time. He is now an Associate Professor of Electrical and Computer Engineering at the University of New Mexico, Albuquerque. His current research includes autonomous attentive perceptual systems and robotics, and he is studying the use of virtual reality technology for the visualization of complex high-performance computing neural simulations. His active research interests include neural networks, fuzzy logic, genetic algorithms, and virtual reality.

He is a Member of the the International Neural Network Society (INNS), the Optical Society of America (OSA), the Association of Computing Machinery (ACM), and Sigma Xi. He is Chairman of the IEEE Neural Networks Council's Technical Committee on Virtual Reality. He is also Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS, an Action Editor for the INNS *Neural Networks Journal*, and is on the Editoral Board of the *International Journal of Virtual Reality*.