Cascade ARTMAP: Integrating Neural Computation and Symbolic Knowledge Processing

Ah-Hwee Tan

Abstract— **This paper introduces a hybrid system termed cascade adaptive resonance theory mapping (ARTMAP) that incorporates symbolic knowledge into neural-network learning and recognition. Cascade ARTMAP, a generalization of fuzzy ARTMAP, represents intermediate attributes and rule cascades of rule-based knowledge explicitly and performs multistep inferencing. A rule insertion algorithm translates if–then symbolic rules into cascade ARTMAP architecture. Besides that initializing networks with prior knowledge can improve predictive accuracy and learning efficiency, the inserted symbolic knowledge can be refined and enhanced by the cascade ARTMAP learning algorithm. By preserving symbolic rule form during learning, the rules extracted from cascade ARTMAP can be compared directly with the originally inserted rules. Simulations on an animal identification problem indicate that *a priori* symbolic knowledge always improves system performance, especially with a small training set. Benchmark study on a DNA promoter recognition problem shows that with the added advantage of fast learning, cascade ARTMAP rule insertion and refinement algorithms produce performance superior to those of other machine learning systems and an alternative hybrid system known as knowledge-based artificial neural network (KBANN). Also, the rules extracted from cascade ARTMAP are more accurate and much cleaner than the NofM rules extracted from KBANN.**

Index Terms—**ARTMAP, hybrid system, promotor recognition, rule extraction, rule insertion, rule refinement.**

I. INTRODUCTION

PRIOR knowledge of a problem domain can help a neural network in learning to solve the problem. Specifically, preexisting symbolic rules can be used to initialize a neural network architecture before learning. Not only can rule insertion improve network learning efficiency, it also serves to provide knowledge that is not captured by training cases or that cannot be easily learned by a neural network, and thus improves the system predictive performance. In addition, incomplete or partially correct symbolic knowledge can be refined or enhanced through neural network learning algorithms. Rule insertion and refinement in neural networks therefore automates symbolic knowledge enhancement and repair.

A popular approach to rule insertion and refinement uses rules to initialize the architecture of a multilayer neural network and refines the network using a backpropagation algorithm [9], [10], [20]. One major problem of using backpropagation (BP) networks to refine rule-based knowledge is the preservation of symbolic knowledge. Under the weight tuning process of a backpropagation algorithm, symbolic rules quickly lose their original meanings. In fact, large shifts in the meanings of hidden units can occur as a result of training [19].

Manuscript received May 16, 1994; revised February 13, 1995, January 8, 1996, and October 8, 1996.

THe author is with the Institute of Systems Science, National University of Singapore, Singapore 119597.

Publisher Item Identifier S 1045-9227(97)01751-7.

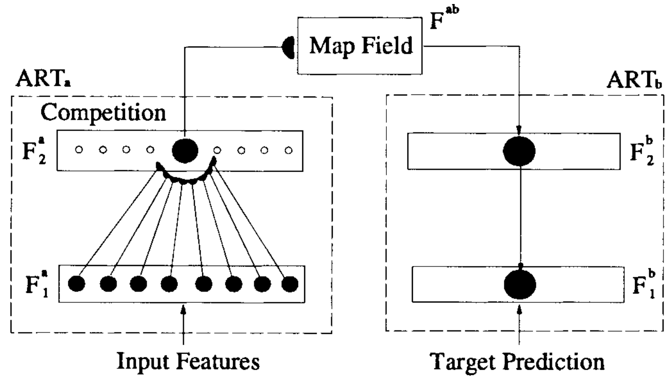

Fig. 1. Rules in fuzzy ARTMAP. Each node in the F_2^a field represents a recognition category of ART_a input patterns. Through the inter-ART map field, each such node is associated to an ART_b category in the F_2^b field, which in turn encodes a prediction. Learned weight vectors, one for each F_2^a node, constitute a set of rules that link antecedents to consequents. The number of rules equals the number of F_2^a nodes that become active during learning.

Another major limitation of the BP approach is that the initial rule base has to be roughly complete, or else the initial network architecture created may not be sufficiently rich for dealing with the problem domain. As the standard backpropagation algorithm is not able to create additional nodes or connections dynamically during learning, a network initialized by a small set of rules may even have a lower chance of eventually learning the task. This problem was noted and partially solved by Lacher *et. al.*, who used virtual rules to create potential connections for learning [10]. However, in general, it is difficult to decide beforehand the virtual rules or connections desired. Tresp *et al.* [21] employed a learning algorithm that allowed creation of basis functions during learning. However, as their model only represents rules associating input attributes to output predictions, the network is not general enough to deal with rule-based domain theories involving intermediate attributes and rule chaining.

This paper introduces a hybrid system called cascade ARTMAP (adaptive resonance theory mapping) that solves the above problems [16], [17]. Cascade ARTMAP, a generalization of fuzzy ARTMAP [3], represents intermediate variables of rule-based knowledge explicitly and performs multistep inferencing (rule chaining). In ARTMAP and cascade ARTMAP systems, the recognition categories learned by the F_2^a category nodes are compatible with rules that

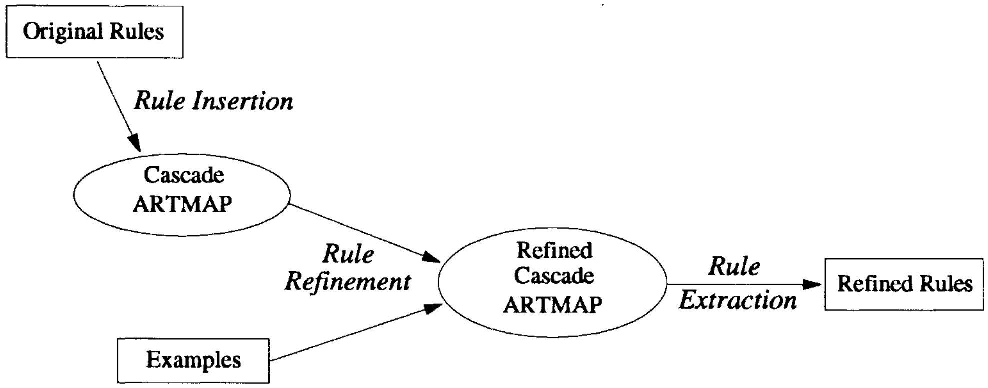

Fig. 2. Cascade ARTMAP for symbolic knowledge refinement and evaluation.

link antecedents to consequents (Fig. 1). Therefore rules can be readily inserted into a cascade ARTMAP network that can then be trained by examples. During learning, new recognition categories (rules) can be created dynamically to cover the deficiency of the domain theory. This is in contrast with the static architecture of the standard slow learning backpropagation networks. Also, by the self-stabilizing property, learning in cascade ARTMAP does not wash away existing knowledge and the meanings of units do not shift. This allows preservation of symbolic rules during neural network learning. Using a generalized ARTMAP rule extraction procedure [6], [7], the final system states can be converted back to a compact set of rules. This enables direct comparison of the original knowledge with the refined rules. Also, each extracted rule is associated with a confidence factor that indicates its importance or usefulness. This allows ranking and evaluation of the extracted knowledge. In all, the cascade ARTMAP rule insertion, refinement, and extraction procedures form a paradigm for symbolic knowledge refinement and evaluation (Fig. 2).

Cascade ARTMAP has been evaluated on two benchmark problems. The first animal identification problem is based on a sample rule-based deductive system [24]. Simulation results show that prior knowledge, even when incomplete, always improves system performance. The effect is most significant when few training cases are available. Benchmark study on the second domain, a DNA promoter recognition problem, compares cascade ARTMAP theory refinement capability with an alternative approach known as the knowledge-based artificial neural network (KBANN) that refines knowledge of feedforward networks using a backpropagation algorithm [18], [20]. Simulation results indicate that with the added advantage of fast learning, the cascade ARTMAP rule insertion and refinement algorithms produce performance superior to that of KBANN. More importantly, the refined rules extracted from cascade ARTMAP are much simpler than the NofM rules extracted from KBANN and can be compared directly with the original symbolic rules.

The remaining sections of this paper are organized as follows. To make this article self-contained, Section II provides a summary of fuzzy ART and fuzzy ARTMAP systems. Section III motivates the generalization of fuzzy ARTMAP and presents the cascade ARTMAP rule insertion, rule chaining, rule refinement, and rule extraction algorithms. The final section illustrates cascade ARTMAP performance on the animal identification and DNA promoter recognition problems.

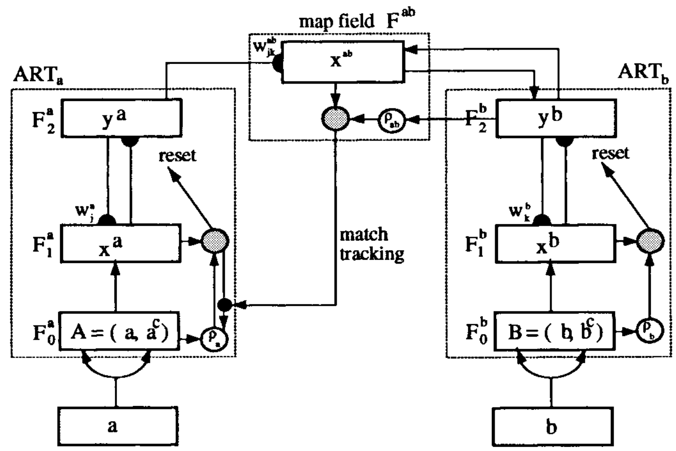

Fig. 3. Fuzzy ARTMAP architecture. The ART_a complement coding preprocessor transforms the M_a-vector $\mathbf{a}$ into the $2M_a$-vector $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$ at the ART_a field F_0^a. $\mathbf{A}$ is the input vector to the ART_a field F_1^a. Similarly, the input to F_1^b is the $2M_b$-vector $(\mathbf{b}, \mathbf{b}^c)$. When ART_b disconfirms a prediction of ART_a, map field inhibition induces the match tracking process. Match tracking raises the ART_a vigilance (ρ_a) to just above the F_1^a-to-F_0^a match ratio $|\mathbf{x}^a|/|\mathbf{A}|$. This triggers an ART_a search which leads to activation of either an ART_a category that correctly predicts $\mathbf{b}$ or to a previously uncommitted ART_a category node.

II. Fuzzy ARTMAP

Fuzzy ARTMAP [3], a generalization of binary ARTMAP [4], learns to classify inputs by a pattern of fuzzy membership values between zero and one indicating the extent to which each feature is present. This generalization is accomplished by replacing the ART 1 modules [2] of the binary ARTMAP system with fuzzy ART modules [5]. Each ARTMAP system includes a pair of ART modules (ART_a and ART_b) that create stable recognition categories in response to arbitrary sequences of input patterns (Fig. 3). An associative learning network and an internal controller link these modules to make the ARTMAP system operate in real time.

A. Fuzzy ART

Fuzzy ART [5] incorporates computations from fuzzy set theory into ART systems. By replacing the crisp (nonfuzzy) intersection operator ($\cap$) that describes ART 1 dynamics [2] by the fuzzy AND operator ($\wedge$) of fuzzy set theory, fuzzy ART can learn stable categories in response to either analog or binary patterns.

ART Field Activity Vectors: Each ART system includes a field F_0 of nodes that represent a current input vector; a field F_1 that receives both bottom-up input from F_0 and top-down input from a field F_2 that represents the active code, or category. Vector $\mathbf{I}$ denotes F_0 activity; vector $\mathbf{x}$ denotes F_1 activity; and vector $\mathbf{y}$ denotes F_2 activity.

Weight Vector: Associated with each F_2 category node j is a vector $\mathbf{w}_j$ of adaptive weights, or long-term memory (LTM) traces. Initially

$$w_{j1}(0) = \cdots = w_{jM}(0) = 1 \tag{1}$$

then each category is *uncommitted*. After a category codes its first input, it becomes *committed*.

Parameters: A choice parameter $\alpha > 0$, a learning rate parameter $\beta \in [0,1]$, and a vigilance parameter $\rho \in [0,1]$ determine fuzzy ART dynamics.

Category choice: For each input $\mathbf{I}$ and F_2 node j, the *choice function* T_j is defined by

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{2}$$

where the fuzzy intersection $\wedge$ [25] is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i) \tag{3}$$

and where the norm $|\cdot|$ is defined by

$$|\mathbf{p}| \equiv \sum_{i=1}^{M} |p_i|. \tag{4}$$

The system makes a *category choice* when at most one F_2 node can become active at a given time. The index J denotes the chosen category, where

$$T_J = \max\{T_j : j = 1 \cdots N\}. \tag{5}$$

When the J^{th} category is chosen, $y_J = 1$; and $y_j = 0$ for $j \neq J$.

Resonance or Reset: Resonance occurs if the *match function* $|\mathbf{I} \wedge \mathbf{w}_J|/|\mathbf{I}|$ of the chosen category meets the vigilance criterion

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho. \tag{6}$$

Learning then ensues, as defined below. Otherwise *Mismatch reset* occurs, where the value of the choice function T_J is set to 0 for the duration of the input presentation . The search process continues until a chosen category J satisfies the matching criterion (6).

Learning: Once search ends, the weight vector $\mathbf{w}_J$ learns according to the equation

$$\mathbf{w}_J^{(\text{new})} = \beta(\mathbf{I} \wedge \mathbf{w}_J^{(\text{old})}) + (1 - \beta)\mathbf{w}_J^{(\text{old})}. \tag{7}$$

Fast learning corresponds to setting $\beta = 1$. Using the fast learning and slow recoding option, we set $\beta = 1$ when J is an uncommitted node and take $\beta < 1$ after the category is committed.

Normalization by complement coding: Normalization of fuzzy ART inputs prevents category proliferation. The complement coded $F_0 \rightarrow F_1$ input $\mathbf{I}$ is the $2M$-dimensional vector

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, \cdots, a_M, a_1^c, \cdots, a_M^c) \tag{8}$$

where

$$a_i^c \equiv 1 - a_i. \tag{9}$$

A complement coded input is automatically normalized, because

$$|\mathbf{I}| = |(\mathbf{a}, \mathbf{a}^c)| = \sum_{i=1}^{M} a_i + (M - \sum_{i=1}^{M} a_i) = M. \tag{10}$$

With complement coding, the initial condition

$$w_{j1}(0) = \cdots = w_{j,2M}(0) = 1 \tag{11}$$

replaces the fuzzy ART initial condition (1).

B. ARTMAP Prediction and Search

Fuzzy ARTMAP incorporates two fuzzy ART modules ART$_a$ and ART$_b$ that are linked together via an inter-ART map field F^{ab}. The map field forms predictive associations between categories and realizes the ARTMAP *match tracking rule*.

ART$_a$ and ART$_b$: Inputs to ART$_a$ and ART$_b$ are complement coded. For ART$_a$, $\mathbf{I} = \mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$; and for ART$_b$, $\mathbf{I} = \mathbf{B} = (\mathbf{b}, \mathbf{b}^c)$ (Fig. 3). For ART$_a$, $\mathbf{x}^a$ denotes the F_1^a output vector; $\mathbf{y}^a$ denotes the F_2^a output vector; and $\mathbf{w}_j^a$ denotes the j^{th} ART$_a$ weight vector. For ART$_b$, $\mathbf{x}^b$ denotes the F_1^b output vector; $\mathbf{y}^b$ denotes the F_2^b output vector; and $\mathbf{w}_k^b$ denotes the k^{th} ART$_b$ weight vector. For the map field, $\mathbf{x}^{ab}$ denotes the F^{ab} output vector, and $\mathbf{w}_j^{ab}$ denotes the weight vector from the $j^{th} F_2^a$ node to F^{ab}.

Map Field Activation: The map field F^{ab} receives input from either or both of the ART$_a$ or ART$_b$ category fields. A chosen F_2^a node J sends input to the map field F^{ab} via the weights $\mathbf{w}_J^{ab}$. An active F_2^b node K sends input to F^{ab} via one-to-one pathways between F_2^b and F^{ab}. If both ART$_a$ and ART$_b$ are active, then F^{ab} remains active only if ART$_a$ predicts the same category as ART$_b$. The F^{ab} output vector $\mathbf{x}^{ab}$ obeys

$$\mathbf{x}^{ab} = \begin{cases} \mathbf{y}^b \wedge \mathbf{w}_J^{ab} & \text{if the } J^{\text{th}} \ F_2^a \text{ node is active and } F_2^b \\ & \quad \text{is active} \\ \mathbf{w}_J^{ab} & \text{if the } J^{\text{th}} \ F_2^a \text{ node is active and } F_2^b \\ & \quad \text{is inactive} \\ \mathbf{y}^b & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is active} \\ \mathbf{0} & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is inactive .} \end{cases} \tag{12}$$

By (12), $\mathbf{x}^{ab} = \mathbf{0}$ if $\mathbf{y}^b$ fails to confirm the map field prediction made by $\mathbf{w}_J^{ab}$. Such a mismatch event triggers an ART$_a$ search for a better category, as follows.

Match Tracking: At the start of each input presentation ART$_a$ vigilance ρ_a equals a baseline vigilance parameter $\overline{\rho_a}$. When a predictive error occurs, match tracking raises ART$_a$

vigilance just enough to trigger a search for a new F_2^a coding node. ARTMAP detects a predictive error when

$$|\mathbf{x}^{ab}| < \rho_{ab}|\mathbf{y}^b| \tag{13}$$

where ρ_{ab} is the map field vigilance parameter. A signal from the map field to the ART$_a$ orienting subsystem causes ρ_a to "track the F_1^a match." That is, ρ_a increases until it is slightly higher than the F_1^a match value $|\mathbf{A} \wedge \mathbf{w}_J^a||\mathbf{A}|^{-1}$. Then, since ART$_a$ fails to meet the matching criterion, the search for another F_2^a node begins.

Map Field Learning: Weights w_{jk}^{ab} in $F_2^a \rightarrow F^{ab}$ paths initially satisfy

$$w_{jk}^{ab}(0) = 1. \tag{14}$$

During resonance with the ART$_a$ category J active, $\mathbf{w}_J^{ab}$ approaches the map field vector $\mathbf{x}^{ab}$ as in (7). With fast learning, once J learns to predict the ART$_b$ category K, that association is permanent; i.e., $w_{JK}^{ab} = 1$ for all time.

III. CASCADE ARTMAP

A. Rule Cascade Representation

ARTMAP handles a class of if–then rules that map a set of input attributes directly to a disjoint set of output attributes. Symbolic rule-based knowledge, on the other hand, often involves *rule cascades* and *intermediate attributes*. A set of rules is said to form a *rule cascade* when a consequent of a rule also serves as an antecedent of another rule. Such attributes that have dual roles are usually called *intermediate attributes* in contrast to *input attributes* that only serve as antecedents and *output attributes* that only serve as consequents. Through intermediate attributes, firing of a rule may lead to the firing of another rule at a later stage of an inferencing process. Intermediate attributes and rule cascades are useful for feature abstraction and task decomposition so that only a small set of simple rules is needed at each level of the cascade.

In this paper, the proposed solution to representing rule cascades is cascade ARTMAP that uses ARTMAP architecture but generalizes the ARTMAP one-step prediction process to multistep inferencing. Cascade ARTMAP unifies the ARTMAP input attribute field F_1^a and output attribute field F_1^b in the sense that both F_1^a and F_1^b contain the input, output, and intermediate attributes. This allows representation of arbitrary mappings from attributes to attributes. Consider the two rules below that form a simple two-level rule cascade

Rule 1: IF A and B THEN C

Rule 2: IF C and D THEN E

where A, B, and D are input attributes, C is an intermediate attribute, and E is an output attribute. All attributes (A, B, C, D, and E) are represented in both F_1^a and F_1^b (Fig. 4). For Rule 1, an F_2^a category node is used to encode A and B, and is associated to an F_2^b node that predicts C. Likewise for Rule 2, an F_2^a node is used to encode C and D, and is associated to an F_2^b node predicting E. By unifying the input field (F_1^a) and the output field (F_1^b) of ARTMAP, several desired features of symbolic processing are obtained. Besides that

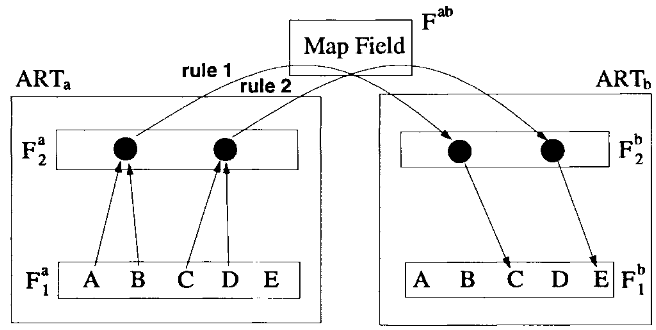

Fig. 4. Cascade ARTMAP representation of the sample rule cascade.

rule insertion can be readily performed in cascade ARTMAP, rule chaining and inferencing can also be performed as in production systems.

B. Rule Insertion

As the knowledge structure of cascade ARTMAP is compatible with rule-based knowledge representation, if–then rules can be readily translated into the recognition categories of a cascade ARTMAP system. Initializing a cascade ARTMAP network with preexisting rules before learning serves to set up the global solution structure. This helps to improve learning efficiency and predictive accuracy. Without rule insertion, cascade ARTMAP performance reduces to that of fuzzy ARTMAP.

Rule insertion proceeds in two phases. The first phase parses all rules for attribute names to set up a *symbol table* in which each attribute in the rules has a unique entry. Based on the symbol table, the second phase translates each rule into two $2M$-dimensional vectors $\mathbf{A}$ and $\mathbf{B}$, where M is the total number of attributes in the symbol table, as inputs to the cascade ARTMAP ART$_a$ and ART$_b$ modules. Given a rule of the following format:

$$\begin{aligned}&\text{IF } x_1, x_2, \cdots, x_m, \neg\bar{x}_1, \neg\bar{x}_2, \cdots, \neg\bar{x}_{\bar{m}}\\ &\text{THEN } \; y_1, y_2, \ldots, y_n, \neg\bar{y}_1, \neg\bar{y}_2, \ldots, \neg\bar{y}_{\bar{n}}\end{aligned} \tag{15}$$

where $x_1, \cdots, x_m$ and $y_1, \ldots, y_n$ are positive attributes, and $\bar{x}_1, \cdots, \bar{x}_{\bar{m}}$ and $\bar{y}_1, \ldots, \bar{y}_{\bar{n}}$ preceded by the logical NOT operator $\neg$ are negative attributes, the algorithm derives the pair of vectors

$$\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) \quad \text{and} \quad \mathbf{B} = (\mathbf{b}, \mathbf{b}^c) \tag{16}$$

such that for each index $j = 1, \cdots, M$

$$(a_j, a_j^c) = \begin{cases}(1,0) & \text{if } e_j = x_i \text{ for some } i \in \{1, \cdots, m\}\\ (0,1) & \text{if } e_j = \bar{x}_i \text{ for some } i \in \{1, \cdots, \bar{m}\}\\ (0,0) & \text{otherwise}\end{cases} \tag{17}$$

and

$$(b_j, b_j^c) = \begin{cases}(1,0) & \text{if } e_j = y_i \text{ for some } i \in \{1, \cdots, n\}\\ (0,1) & \text{if } e_j = \bar{y}_i \text{ for some } i \in \{1, \cdots, \bar{n}\}\\ (0,0) & \text{otherwise}\end{cases} \tag{18}$$

where e_j is the jth attribute in the symbol table. Note that complement coding is employed above for encoding both

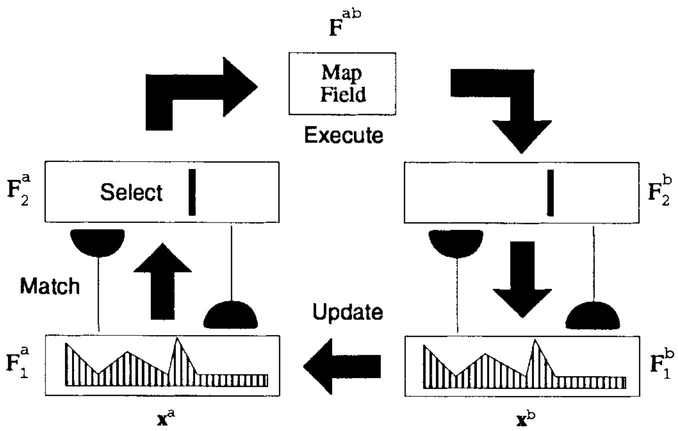

Fig. 5. Cascade ARTMAP rule chaining and inferencing: $\mathbf{x}^a$ represents the system's memory state and accumulates attribute values during multistep inferencing.

the positive and negative attributes. If the rules contain no negative attribute, the complement vectors $\mathbf{a}^c$ and $\mathbf{b}^c$ may be eliminated.

The vector pairs derived from the rules are then used as training patterns to initialize a cascade ARTMAP network. The network learning and inferencing algorithms will be described in subsequent sections. It suffices to note at this point that each distinct vector $\mathbf{A}$ is translated into a recognition category in ART$_a$ and likewise each distinct vector $\mathbf{B}$ is translated into a recognition category in ART$_b$. Given a pair of vectors $\mathbf{A}$ and $\mathbf{B}$ derived from a rule, their respective recognition categories are associated through the map field. During network initialization, the vigilance parameters ρ_a and ρ_b are each set to one to ensure that only identical attribute vectors are grouped into one recognition category. Contradictory symbolic rules are detected during rule insertion when identical input attribute vectors are associated with distinct output attribute vectors. The detection is achieved through a *perfect mismatch* phenomenon, in which the system tries to raise ART$_a$ vigilance ρ_a above one in response to a mismatch in ART$_b$ (Section III-E).

C. Rule Chaining and Inferencing

A symbolic production rule system consists of three main components: a *working memory* component, a *rule* or *production* component, and an external inference engine or *interpreter*. The interpreter repeatedly performs a three-phase cycle, consisting of a *match* phase, a *select* phase, and an *execute* phase. In the match phase, the interpreter compares the antecedent set of each rule against the working memory content. Rules with completely matched antecedents are included into a *conflict* set. In the select phase, a single rule is selected from the conflict set using some strategies. If the conflict set is empty, the system halts. Otherwise, in the execute phase, the interpreter adds the consequent(s) of the selected rule to the working memory.

In cascade ARTMAP, the attribute fields F_1^a and F_1^b can be identified as the working memory component (Fig. 5). F_1^a maintains the current memory state $\mathbf{x}^a$ and provides the antecedents for condition matching and rule firing. F_1^b stores the next memory state $\mathbf{x}^b$ derived through a rule firing. The rule component is implemented by the two category fields F_2^a and F_2^b, the map field F^{ab}, and their interconnections. The match, select, and execute three-phase cycle is performed without an external interpreter. In the match phase, a choice function T_j^a is computed for each F_2^a category node (rule) based on the memory state $\mathbf{x}^a$. With parallel implementation, the match phase can be performed in a single activation process. The select phase is realized by a winner-take-all interaction among all F_2^a nodes in which the F_2^a node with the largest choice function T_j^a is identified. If the selected node (rule) does not satisfy the ART$_a$ vigilance constraint, the system goes through another round of memory search to select another F_2^a node that satisfies the ART$_a$ vigilance criterion. If no such node exists, the system halts. Otherwise, in the execute phase, the consequent(s) of the selected rule is(are) read out into F_1^b. Note that exact match is not required for a rule to be fired as long as it satisfies the ART$_a$ vigilance criterion. At the end of the cycle, the new memory state $\mathbf{x}^b$ is used to update $\mathbf{x}^a$ in F_1^a to prepare for the next inferencing cycle. For the sample rule cascade (Fig. 4), the input attribute set {A,B,D} activates F_2^a node J_1 that infers C. Through the memory update process, C is fed back from F_1^b to F_1^a. The memory state $\mathbf{x}^b$ which contains {A,B,C,D} then activates J_2 in the next inferencing cycle that derives E.

D. Learning and Rule Refinement

Learning in cascade ARTMAP is more complicated than that in fuzzy ARTMAP as a chain of rule firing is involved in making a prediction. The proposed solution is a backtracking algorithm that identifies all rules (F_2^a nodes) responsible for making a prediction by tracing from the last rule fired. Specifically, if J is the last F_2^a node selected which makes the prediction, the algorithm identifies a *precursor* set $\Psi(J)$ that contains node J and all F_2^a nodes that result in the firing of node J. The backtracking occurs in the direction of $F_2^a \rightarrow F_1^a \rightarrow F_1^b \rightarrow F_2^b \rightarrow F^{ab} \rightarrow F_2^a$. For example in Fig. 4, the backtracking algorithm traces from J_2 in F_2^a to its antecedents {C,D} in F_1^a. It then checks that C in F_1^b is an intermediate attribute activated by K_1 in F_2^b, and finally traces to J_1 in F_2^a. The backtracking stops at J_1 as its antecedents are all input attributes. The precursor set of the F_2^a node J_2, $\Psi(J_2)$ is thus evaluated to be $\{J_1, J_2\}$.

If the prediction made by node J is correct, for each F_2^a node (rule) j in the precursor set $\Psi(J)$, the weight vector (antecedent set) $\mathbf{w}_j^a$ is reduced toward its fuzzy intersection with the F_1^a activity vector $\mathbf{x}^a$. In the binary pattern and fast learning case, a fired rule learns to ignore those features that are absent in the current input. This results in a generalization by reducing the number of features the rule attends to.

A more complicated situation occurs when a prediction error is encountered. With a long chain of rule firing, blame assignment can be difficult as it is unclear which rule in the inferencing path causes the error. To handle prediction mismatch, a *mini-match tracking* process raises the ART$_a$ vigilance ρ_a by slightly more than the *minimum* match achieved by the fired rules. Mini-match tracking is equivalent to the

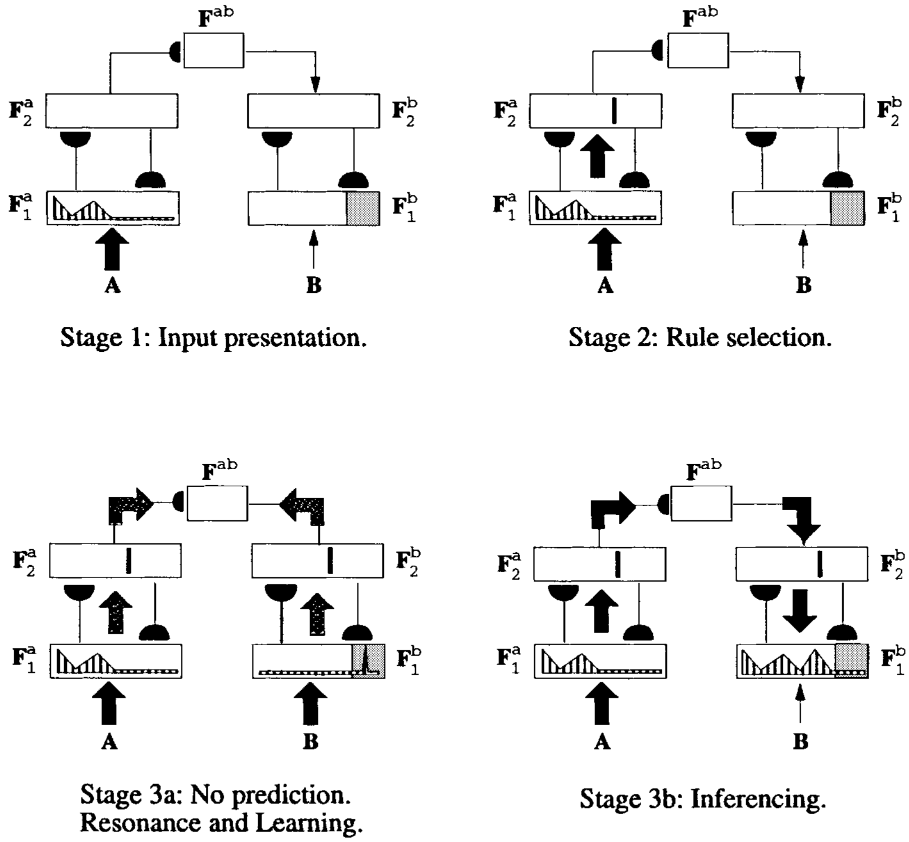

Fig. 6. Cascade ARTMAP algorithm stages 1, 2, and 3. The shaded subfields of F_1^b represent output attributes.

parallel match tracking mechanism used in fusion ARTMAP [1]. This method inhibits the F_2^a node with the minimum match from firing again for the current input. The assumption is that the rule with the worst match is most likely to be the one which causes the prediction error. The system then goes through another round of memory search and inferencing with a higher vigilance until a resonance is achieved.

E. Cascade ARTMAP Algorithm

As an on-line real-time system, cascade ARTMAP needs not separate learning and performance phases, i.e., the system functions in response to the current input environment. Given an F_1^a input vector, cascade ARTMAP undergoes a series of prediction loops until either an uncommitted F_2^a node is selected (which means no prediction), or a correct prediction is made by a committed F_2^a node (as in fuzzy ARTMAP). In each prediction loop, cascade ARTMAP accumulates intermediate attribute values through a series of inferencing cycles until one or more output attributes are derived. The cascade ARTMAP dynamics, as illustrated in Figs. 6 and 7, are formalized below. An *A then B* paradigm is used in which the ART_a input vector $\mathbf{A}$ is processed before the ART_b input vector $\mathbf{B}$.

Activity Vectors: Let $\mathbf{A}$ and $\mathbf{B}$ denote the F_1^a and F_1^b input vectors, respectively. Let $\mathbf{x}^a \equiv (\mathbf{x}_i^a, \mathbf{x}_h^a, \mathbf{x}_o^a, \mathbf{x}_i^{ac}, \mathbf{x}_h^{ac}, \mathbf{x}_o^{ac})$ and $\mathbf{x}^b \equiv (\mathbf{x}_i^b, \mathbf{x}_h^b, \mathbf{x}_o^b, \mathbf{x}_i^{bc}, \mathbf{x}_h^{bc}, \mathbf{x}_o^{bc})$ denote the $2M$-dimensional F_1^a and F_1^b activity vectors, respectively, where $\mathbf{x}_i^a$ and $\mathbf{x}_i^b$ denote the M_i-dimensional input attribute vectors; $\mathbf{x}_h^a$ and $\mathbf{x}_h^b$ denote the M_h-dimensional intermediate attribute vectors; $\mathbf{x}_o^a$ and $\mathbf{x}_o^b$ denote the M_o-dimensional output attribute vectors; and $\mathbf{x}_i^{ac}$, $\mathbf{x}_i^{bc}$, $\mathbf{x}_h^{ac}$, $\mathbf{x}_h^{bc}$, $\mathbf{x}_o^{ac}$, and $\mathbf{x}_o^{bc}$ denote the respective complement attribute vectors. $\mathbf{x}^a$ and $\mathbf{x}^b$ are also known as memory state vectors. Let $\mathbf{y}^a$ and $\mathbf{y}^b$ denote the F_2^a and F_2^b activity vectors, respectively. Let $\mathbf{x}^{ab}$ denote the map field F^{ab} activity vector.

Weight Vectors: Let $\mathbf{w}_j^a$ and $\mathbf{w}_j^b$ denote the $2M$-dimensional weight vectors of the j^{th} category node in F_2^a and F_2^b, respectively. Let $\mathbf{w}_j^{ab}$ denote the weight vector from the jth F_2^a node to F^{ab}. Initially, the weight vectors contain all "1's". This implies that all category nodes are uncommitted and all F_2^a nodes are not associated with any prediction.

Scope Vectors: Let $\mathbf{S}_j$ denote the $2M$-dimensional scope vector of the j^{th} category node in F_2^a. A scope vector identifies the attributes relevant to an F_2^a node and allows a more accurate computation of its match function. For an uncommitted F_2^a node j, the scope vector $\mathbf{S}_j \equiv (\mathbf{s}_j, \mathbf{s}_j)$ is defined by

$$s_{ji} = \begin{cases} 1 & \text{if } i \text{ indexes an input attribute} \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

For an F_2^a node j created by an inserted rule, the scope vector $\mathbf{S}_j \equiv (\mathbf{s}_j, \mathbf{s}_j)$ is defined by

$$s_{ji} = \begin{cases} 1 & \text{if } i \text{ indexes an attribute at a level previous to} \\ & \text{the rule's consequent(s) in the rule cascade} \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

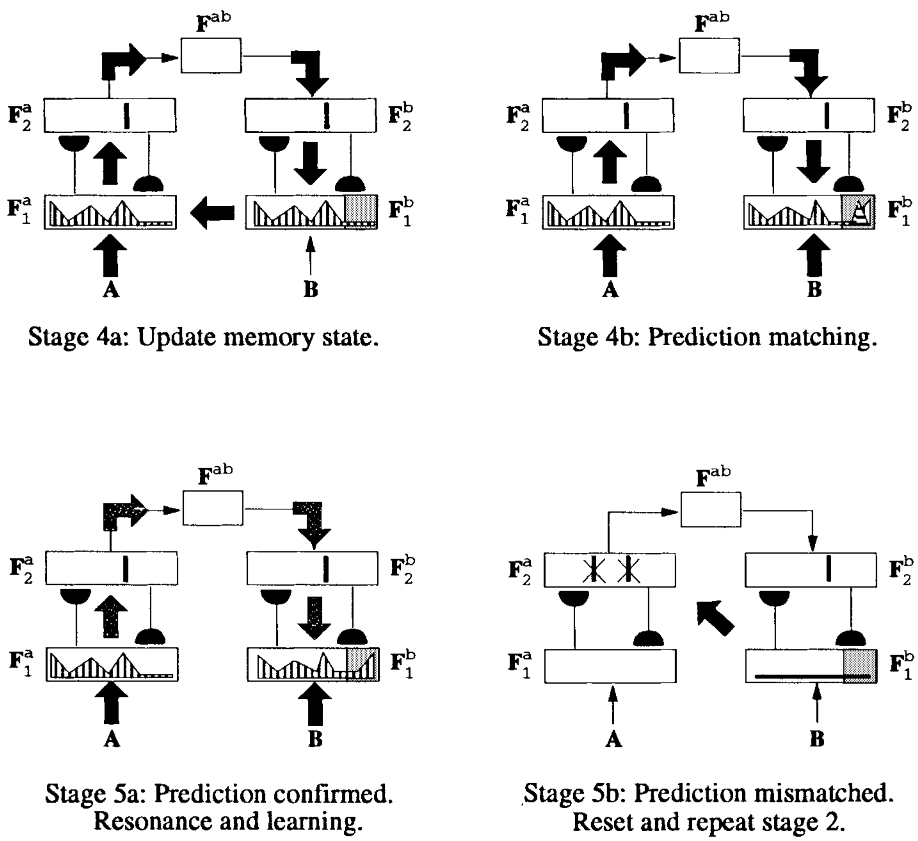

Fig. 7. Cascade ARTMAP algorithm stages 4 and 5. The shaded subfields of F_1^b represent output attributes.

Parameters: Cascade ARTMAP dynamics are determined by the choice parameters $\alpha_a > 0$ and $\alpha_b > 0$; the learning rates $\beta_a \in [0,1]$ and $\beta_b \in [0,1]$; and the vigilance parameters $\rho_a \in [0,1]$ and $\rho_b \in [0,1]$. During network initialization, the network learns the patterns derived from rules (Section III-B) using $\rho_a = \rho_b = 1$ such that each distinct attribute vector creates a category. During network refinement, the system learns example patterns using $\rho_b = 1$ for output classification and $\rho_a < 1$ to allow input generalization.

Stage 1 (Input Presentation): At the beginning of an input presentation, the ART$_a$ vigilance ρ_a equals a baseline vigilance value $\bar{\rho_a}$. F_1^a contains the input vector $\mathbf{A}$

$$\mathbf{x}^a = \mathbf{A}. \tag{21}$$

Stage 2 (Rule Selection): Given the memory state vector $\mathbf{x}^a$, for each F_2^a node j, the choice function T_j^a is defined by

$$T_j^a = \frac{|\mathbf{x}^a \wedge \mathbf{w}_j^a|}{\alpha_a + |\mathbf{w}_j^a|} \tag{22}$$

where the fuzzy AND operator $\wedge$ is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i) \tag{23}$$

and the norm $|.|$ is defined by

$$|\mathbf{p}| = \sum_i |p_i| \tag{24}$$

for vectors $\mathbf{p}$ and $\mathbf{q}$. The system is said to make a *category choice* when at most one F_2^a node can become active at a given time. The category choice is indexed at J, where

$$T_J^a = \max\{T_j^a : \text{for all } F_2^a \text{ node } j\ \}. \tag{25}$$

If more than one T_j^a is maximal, the F_2^a category node j with the smallest index is chosen. In particular, nodes become committed in order $j = 1, 2, 3, \cdots$. When the J^{th} category is chosen, $y_J^a = 1$; and $y_j^a = 0$ for $j \neq J$.

Resonance occurs if the match function m_J^a of the selected node J meets the vigilance criterion

$$m_J^a = \frac{|\mathbf{x}^a \wedge \mathbf{w}_J^a \wedge \mathbf{S}_J|}{|\mathbf{x}^a \wedge \mathbf{S}_J|} \geq \rho_a \tag{26}$$

where the generalized fuzzy AND operation $\wedge$ is defined by

$$(\wedge_{j=1}^N \mathbf{p}_j)_i \equiv \min(p_{1i}, \ldots, p_{Ni}) \tag{27}$$

for vectors $\mathbf{p}_1, \cdots, \mathbf{p}_N$, and the norm $|.|$ is defined as in (24). Otherwise mismatch reset occurs in which the value of the choice function T_J^a is set to zero for the duration of the input presentation to prevent persistent selection of the same category during search. Stage 2 then repeats to select another new index J.

Stage 3a (No Prediction): If the selected F_2^a node J has no prediction, i.e.,

$$w_{Jk}^{ab} = 1 \ \text{ for all } F^{ab} \text{ node } k \tag{28}$$

each F_2^a node j in the precursor set $\Psi(J)$ learns the F_1^a activity pattern according to the equation

$$\mathbf{w}_j^{a\ (\text{new})} = (1-\beta_a)\mathbf{w}_j^{a\ (\text{old})} + \beta_a(\mathbf{x}^a \wedge \mathbf{w}_j^{a\ (\text{old})}). \tag{29}$$

If the input vector $\mathbf{B}$ is present, a category node is selected in F_2^b as in stage 2. The selected F_2^b node K learns the F_1^b pattern according to the equation

$$\mathbf{w}_K^b{}^{(\text{new})} = (1-\beta_b)\mathbf{w}_K^b{}^{(\text{old})} + \beta_b(\mathbf{x}^b \wedge \mathbf{w}_K^b{}^{(\text{old})}). \quad (30)$$

The F_2^a category node J is then associated to the F_2^b node K through the inter-ART map field

$$w_{Jk}^{ab} = \begin{cases} 1 & \text{if } k = K \\ 0 & \text{otherwise.} \end{cases} \quad (31)$$

After that, the system halts.

Stage 3b (Inferencing): If the selected F_2^a node J has learned to make a prediction, i.e., (28) does not hold, its weight vector $\mathbf{w}_J^{ab}$ activates F^{ab}. The F^{ab} activity vector $\mathbf{x}^{ab}$ is defined by

$$\mathbf{x}^{ab} = \mathbf{w}_J^{ab}. \quad (32)$$

Once the map field is active, F_2^b is activated through the one-to-one pathway between F^{ab} and F_2^b . For each F_2^b node k, the choice function T_k^b is defined by

$$T_k^b = x_k^{ab}. \quad (33)$$

The system again makes a category choice indexed at K where

$$T_K^b = \max\{T_k^b\text{: for all } F_2^b \text{ node } k\ \}. \quad (34)$$

When the K^{th} category is chosen, $y_K^b = 1$, and $y_k^b = 0$ for $k \neq K$. The activated F_2^b node K then performs a top–down priming on F_1^b

$$\mathbf{x}^b = \mathbf{w}_K^b. \quad (35)$$

When F_1^b is activated by a category choice in F_2^b, the termination condition is checked by computing a goal signal g

$$g = \sum_{i=1}^{M_o}(x_{oi}^b + x_{oi}^{bc}). \quad (36)$$

A conclusion is reached whenever any output attribute is made known, i.e., $g > 0$.

Stage 4a (Update Memory State): If a conclusion is not reached, i.e., $g = 0$, the memory state vector $\mathbf{x}^a$ is updated with $\mathbf{x}^b$ by the equation

$$\mathbf{x}^a{}^{(\text{new})} = \mathbf{x}^a{}^{(\text{old})} \vee \mathbf{x}^b{}^{(\text{old})} \quad (37)$$

where the fuzzy OR operation $\vee$ is defined by

$$(\mathbf{p} \vee \mathbf{q})_i \equiv max(p_i, q_i) \quad (38)$$

for vectors $\mathbf{p}$ and $\mathbf{q}$. The inferencing cycle then repeats from stage 2.

Stage 4b (Prediction Matching): If a conclusion is reached, i.e., $g > 0$, the match function m_K^b of the prediction $\mathbf{x}^b$ and the F_1^b input vector $\mathbf{B}$ is computed by

$$m_K^b = \frac{|\mathbf{B} \wedge \mathbf{x}^b|}{|\mathbf{B}|}. \quad (39)$$

Stage 5a (Resonance): If the prediction match satisfies the ART$_b$ vigilance criterion ($m_K^b \geq \rho_b$), resonance occurs. The activated F_2^a and F_2^b nodes learn the template patterns in their respective modules as in (29) and (30), respectively. After learning, the system halts.

Stage 5b (Match Tracking): A prediction mismatch triggers a match tracking process. Using mini-match tracking, a node j is identified which has the minimum match function value among all nodes in $\Psi(J)$. The choice function T_j^a of the node j is set to zero during the input presentation. The ART$_a$ vigilance ρ_a is raised to slightly greater than the match achieved by the node j_m

$$\rho_a{}^{(\text{new})} = \max\{\rho_a{}^{(\text{old})}, \min\{m_j^a | j \in \Psi(J)\} + \epsilon\}. \quad (40)$$

Perfect mismatch occurs when the system attempts to increase ρ_a above one. A perfect match in ART$_a$ ($\rho_a = 1$) with a ART$_b$ mismatch indicates the existence of contradictory knowledge where identical antecedent sets are associated with different consequents. After match tracking, a new prediction loop then repeats from stage 2.

F. Rule Extraction

As a direct generalization of fuzzy ARTMAP, cascade ARTMAP architecture can be readily translated into a set of symbolic rules using a generalized ARTMAP rule extraction procedure [6], [7]. A rule pruning procedure selects a small set of rules from cascade ARTMAP networks based on their confidence factors. To derive concise rules, an antecedent pruning procedure aims to remove antecedents from rules while preserving accuracy.

1) Rule Pruning: The rule pruning algorithm derives a confidence factor for each F_2^a category node in terms of its usage frequency in a *training* set and its predictive accuracy on a *predicting* set. As cascade ARTMAP generalizes ARTMAP one-step prediction process to multistep inferencing, an input pattern makes use of a set of F_2^a category nodes in cascade ARTMAP in contrast to a single F_2^a node in fuzzy ARTMAP. For evaluating *usage* and *accuracy*, each F_2^a category node j maintains three counters: an encoding counter c_j, that records the number of training patterns encoded by node j, a predicting counter p_j that records the number of predicting set patterns predicted by node j, and a success counter s_j, that records the number of predicting set patterns predicted correctly by node j.

For each training pattern, the encoding counter (c_j) of each F_2^a node j in the precursor set $\Psi(J)$, where J is the last F_2^a node (rule) fired that makes the prediction, is increased by one. For each predicting set pattern, the predicting counter (p_j) of each F_2^a node j in the precursor set $\Psi(J)$ is increased by one. If the prediction is correct, the success counter (s_j) of each F_2^a node j in the precursor set $\Psi(J)$ is increased by 1. Based on the encoding, predicting, and success counter values, the *usage* (U_j) and the accuracy (A_j) of an F_2^a node j are computed by

$$U_j = c_j / \max\{c_k\text{: for all } F_2^a \text{ node } k\ \} \quad (41)$$

and

$$A_j = P_j / \max\{P_k\text{: for all } F_2^a \text{ node } k\ \} \quad (42)$$

where P_j, the percent of the predicting set pattern predicted correctly by node j, is computed by

$$P_j = s_j / p_j. \tag{43}$$

U_j and A_j are then used to compute the confidence factor of node j by the equation

$$CF_j = \gamma U_j + (1 - \gamma) A_j \tag{44}$$

where $\gamma \in [0, 1]$ is a weighting factor. After confidence factors are determined, recognition categories can be pruned from the network using one of following strategies.

Threshold Pruning: This is the simplest type of pruning where the F_2^a nodes with confidence factors below a given threshold τ are removed from the network. A typical setting for τ is 0.5. This method is fast and provides a first cut elimination of unwanted nodes. To avoid over-pruning, it is sometimes useful to specify a minimum number of recognition categories to be preserved in the system.

Local Pruning: Local pruning removes recognition categories one at a time from an ARTMAP network. The baseline system performance on the training and the predicting sets is first determined. Then the algorithm deletes the recognition category with the lowest confidence factor. The category is replaced, however, if its removal degrades system performance on the training and predicting sets.

A variant of the local pruning strategy updates baseline performance each time a category is removed. This option, called *hill-climbing*, gives slightly larger rule sets but better predictive accuracy. A hybrid strategy first prunes the ARTMAP systems using threshold pruning and then applies local pruning on the remaining smaller set of rules.

2) Antecedent Pruning: During rule extraction, a nonzero weight to an F_2^a category node translates into an antecedent in the corresponding rule. The antecedent pruning procedure calculates an error factor for each antecedent in each rule based on its performance on the training and predicting sets. When a rule (F_2^a node) J makes a prediction error, for each F_2^a node j in the precursor set $\Psi(J)$, each antecedent of the rule j that also appears in the current memory state has its error factor increased in proportion to the smaller of its magnitudes in the rule and in the memory state vector $\mathbf{x}^a$. After the error factor for each antecedent is determined, a local pruning strategy, similar to the one for rules, removes redundant antecedents.

IV. Experimental Results

A. Animal Identification

A sample rule-based deductive problem [24] is used as the first test bed of cascade ARTMAP. The simple domain provides a noise-free closed-world environment to understand and illustrate the system behaviors. The 15-rule knowledge base (Table I) distinguishes among seven types of animals based on 21 input attributes. Four intermediate attributes: mammal, bird, carnivore, and ungulate, are involved in deducing the output animal type. The rules combine to form a three-level rule cascade.

In cascade ARTMAP experiments, the deductive knowledge base serves two functions. Incomplete versions of the rule set are used to initialize the cascade ARTMAP network. The rule set is also used to generate examples for training cascade ARTMAP. 1000 training examples are generated by assigning random values to the input attributes and deriving the output attribute values based on the complete set of 15 rules. The internal reasoning of the rule-based system is thus hidden from the generated training examples which contain only the 21 input attributes and the seven output attributes. A filtering process is applied to remove inconsistent cases such as those in which "does not fly" and "is a good flyer" are both true.

The baseline performance of cascade ARTMAP without rule insertion is first evaluated. The system is trained on a portion of the generated 1000 cases and tested on the remaining patterns. Table II shows that cascade ARTMAP learns the untold regularities almost perfectly, given sufficient data. On the average, 16.8 rules, about the size of the original rule set, are created from 900 training cases.

In the rule insertion simulations, cascade ARTMAP network is initialized with partial rule sets, and then trained and tested as in the previous experiments. The first set of experiments initializes cascade ARTMAP with a five-rule set that recognizes birds. The five-rule set, by itself, only classifies correctly 56.4% of the 1000 test patterns. Through rule refinement, the performance improves significantly to 96.8% given only 100 training examples, and to 99.8% with 900 examples. Moreover, the total number of rules formed in each simulation is fewer than the original 15-rule set that generates the examples. cascade ARTMAP with the five-rule set also performs consistently better than its counterpart without rules. The effect is best observed when the system is trained with a small data set (from 92.3% to 96.8%). The performance gap between the two systems closes up as the number of training pattern increases. This indicates that given sufficient examples and in a noise free environment, cascade ARTMAP is able to derive accurate rules by learning from examples alone.

A similar improvement in performance is observed when the system is initialized with a ten-rule set. However, the performance with the ten-rule set is not better than that with the five-rule set. A careful study reveals that the additional five rules (Z1,Z2,Z5,Z11,Z12) are all isolated rules that do not form a complete inferencing chain, and are thus inactive. In fact, the performance of the ten-rule set alone is exactly the same as that of the five-rule set alone (56.4%). The inactive rules also have a negative effect on the rule refinement process. Given a small set of 100 examples, cascade ARTMAP with the ten-rule set is worse than that with the five-rule set. As the system learns sufficient rules from larger sets of examples, the effect of the inactive rules is eliminated.

The final set of the experiments is on the insertion and refinement of a 13-rule set (all rules except Z10 and Z14) that classifies correctly 76.6% of the test patterns. When initialized with the 13-rule set, cascade ARTMAP consistently discovers variants of the missing rules and achieves 100% correct prediction. Table III shows a sample set of rules discovered. Rule R1 corresponds exactly to Z14 as "has feathers" implies

TABLE I
A SAMPLE KNOWLEDGE BASE THAT IDENTIFIES ANIMALS

Z1	If	?x has hair
	then	?x is a mammal
Z2	If	?x gives milk
	then	?x is a mammal
Z3	If	?x has feathers
	then	?x is a bird
Z4	If	?x flies
	and	?x lays eggs
	then	?x is a bird
Z5	If	?x is a mammal
	and	?x eats meat
	then	?x is a carnivore
Z6	If	?x is a mammal
	and	?x has pointed teeth
	and	?x has claws
	and	?x has forward-pointing eyes
	then	?x is a carnivore
Z7	If	?x is a mammal
	and	?x has hoofs
	then	?x is an ungulate
Z8	If	?x is a mammal
	and	?x chews cud
	then	?x is an ungulate
Z9	If	?x is a carnivore
	and	?x has tawny color
	and	?x has dark spots
	then	?x is a cheetah
Z10	If	?x is a carnivore
	and	?x has tawny color
	and	?x has black stripes
	then	?x is a tiger
Z11	If	?x is an ungulate
	and	?x has long legs
	and	?x has long neck
	and	?x has tawny color
	and	?x has dark spots
	then	?x is a giraffe
Z12	If	?x is an ungulate
	and	?x has white color
	and	?x has black stripes
	then	?x is a zebra
Z13	If	?x is a bird
	and	?x does not fly
	and	?x has long legs
	and	?x has long neck
	and	?x is black and white
	then	?x is a ostrich
Z14	If	?x is a bird
	and	?x does not fly
	and	?x swims
	and	?x is black and white
	then	?x is a penguin
Z15	If	?x is a bird
	and	?x is a good flyer
	then	?x is an albatross

TABLE II
PERFORMANCE OF CASCADE ARTMAP WITH AND WITHOUT RULE INSERTION ON THE ANIMAL IDENTIFICATION PROBLEM. THE SIMULATION RESULTS ARE AVERAGED OVER TEN RUNS

Systems	Train/ Test	Training Iterations	# Nodes /Rules	Predictive Accuracy Correct (%)	SD
Cascade ARTMAP without rules	100/900	2.2	13.4	92.3	1.6
(≡ Fuzzy ARTMAP)	500/500	2.6	16.9	98.9	0.8
	900/100	2.4	16.8	99.6	0.5
5 rules alone	0/1000	-	5	56.4	-
Cascade ARTMAP with 5 rules	100/900	2.2	11.0	96.8	1.7
(Z3, Z4, Z13, Z14, Z15)	500/500	2.3	13.0	99.3	0.3
	900/100	2.3	13.7	99.8	0.4
10 rules alone	0/1000	-	10	56.4	-
Cascade ARTMAP with 10 rules	100/900	2.2	14.3	95.7	1.9
(Z1-Z5, Z11-Z15)	500/500	2.2	14.6	99.4	0.6
	900/100	2.0	15.0	99.8	0.4
13 rules alone	0/1000	-	13	76.6	-
Cascade ARTMAP with 13 rules	100/900	2.0	15.0	99.2	0.4
(less Z10 and Z14)	500/500	2.0	15.0	100.0	0.0
	900/100	2.0	15.0	100.0	0.0

TABLE III
SAMPLE RULES DISCOVERED BY CASCADE ARTMAP DURING LEARNING. R1 ≡ Z14 AND R2 ≡ Z10

R1	**If**	**?x has feathers**	**R2**	**If**	**?x has tawny color**
	and	**?x does not fly**		**and**	**?x has black stripes**
	and	**?x swims**		**then**	**?x is a tiger**
	and	**?x is black and white**			
	then	**?x is a penguin**			

"is a bird" by Z3. Rule R2 is a generalization over the missing rule Z10, as "black stripes" and "tawny color" are sufficient to identify "tiger" in this problem.

B. DNA Promoter Recognition

Promoters are short nucleotide sequences that occur before genes and serve as binding sites for the enzyme RNA polymerase during gene transcription. Identifying promoters is thus an important step in locating genes in DNA sequences. One major approach to DNA matching or sequence comparison concerns with the alignment of DNA sequences. Sequence alignment is usually performed by computing a match function that rewards matches and penalizes mismatches, insertions, and deletions [22], [23]. This can be done by dynamic programming which can be computationally expensive for multiple sequences. Consensus sequence analysis solves the problem of aligning multiple sequences by identifying functionally important sequence features that are conserved in the DNA sequences. For example, consensus patterns of promoter sequences can be identified at the protein binding sites. Besides statistical methods reported in the biological literature, machine learning and information theoretic techniques are also being used for DNA matching and recognition [8], [11]. In this paper, the performance of cascade ARTMAP is compared with an *ad hoc* partial pattern matching algorithm, based on consensus pattern analysis, of which published result on promoter recognition is available [14]. We also make performance comparison to many machine learning systems including backpropagation neural network, ID-3 symbolic learning algorithm, and K-nearest neighbor (KNN) system.

The promoter data set [12] used in the cascade ARTMAP experiments consists of 106 patterns, half of which are positive instances (promoters). Although larger sets of promoter data are available, this version of the promoter data set is used here to allow a direct comparison with the results of the others. Each DNA pattern represents a 57-position window, with the leftmost 50 window positions labeled -50 to -1 and the rightmost seven labeled 1 to 7 (Fig. 8). Each position is a nominal feature which takes one of the four nucleotide values {A, G, T, C}. There is no missing feature value. Using local representation, each 57-position pattern is expanded into a 228-bit nucleotide-position string.

The promoter data set and an imperfect domain theory were used to evaluate a hybrid learning system called KBANN [20]. The imperfect domain theory (Table IV), if requires exact match, only classifies half of the 106 cases correctly. The KBANN theory refinement procedure translates the imperfect theory into a feedforward network, adds links to make the network fully connected between layers, and trains the network using a backpropagation algorithm. Simulation results showed that by incorporating the domain theory, KBANN outperformed many learning/recognition systems, including consensus sequence analysis [14], KNN, ID-3 symbolic learning algorithm [15], and backpropagation network trained purely from examples [20] (Table V).

In cascade ARTMAP experiments, the first two rules of the domain theory are combined into a single rule:

promoter :- conformation, minus_35, minus_10.

Besides providing a slight improvement in system predictive accuracy, the elimination of attribute *contact* reduces cascade ARTMAP network complexity and produces simpler rule sets.

Cascade ARTMAP simulation is performed with parameter values $\alpha_a = \alpha_b = 2$ and $\beta_a = \beta_b = 1$, determined empirically. The input patterns are not complement coded as they already have a uniform norm of 57. In each simulation, cascade ARTMAP is initialized with the domain theory, trained on 96 patterns selected randomly, and tested on the remaining ten patterns. To use a voting strategy, cascade ARTMAP is trained in several simulation runs using different orderings of the training set. For each test case, voting across 20 runs yields a final prediction. An averaging technique similar to voting was also used in the KBANN system [20].

Table V compares the performance of fuzzy ARTMAP and cascade ARTMAP, averaged over 20 simulations, with other alternative systems. Among the systems that do not incorporate *a priori* symbolic knowledge, fuzzy ARTMAP (cascade ARTMAP without rule insertion) achieves the lowest error rate. While the KBANN system and cascade ARTMAP both obtain significant improvement in predictive performance by incorporating rules, cascade ARTMAP produces a lower error rate than KBANN. In addition to the 13 inserted rules, an average of 15. 9 recognition nodes (rules) are created.

In each simulation, rules are also extracted from the trained cascade ARTMAP network. Due to the small data set size, confidence factors are computed solely based on *usage*. Threshold pruning with threshold $\tau = 0.01$ is applied, followed by the rule and antecedent pruning procedures using the local pruning strategy. Comparing predictive performance, rules extracted from cascade ARTMAP are still slightly more accurate than the NofM rules extracted from KBANN [18], [19]. While the cascade ARTMAP rule sets contain more rules than the NofM rule sets, the number of antecedents is almost half of that of the NofM rule sets (Table V).

The promoter rules formulated by cascade ARTMAP are similar in form to the consensus sequences derived by conventional statistical methods. However, whereas consensus sequences are used with an exact match condition, cascade ARTMAP rules are based on competitive activation and do not require exact match in antecedents. Through the approximate matching property, the number of nucleotides used to identify a promoter is usually small (at most four in this case). By contrast, the consensus sequences, obtained by noting the positions with the same base in greater than 50% of the promoter patterns [13], used a minimum of 12 nucleotides.

Table VI shows a sample set of refined promoter rules extracted from cascade ARTMAP. *Conformation* has been

TABLE IV
A RULE-BASED THEORY FOR CLASSIFYING PROMOTERS. IT CONSISTS OF 14 RULES AND A TOTAL OF 83 ANTECEDENTS. THE ANTECEDENT NOTATION T@-36 INDICATES THE NUCLEOTIDE VALUE T IN POSITION -36

promoter	:-	**conformation, contact.**
contact	:-	minus_35, minus_10.
minus_35	:-	C@-37, T@-36, T@-35, G@-34, A@-33, C@-32.
minus_35	:-	T@-36, T@-35, G@-34, C@-32, A@-31.
minus_35	:-	T@-36, T@-35, G@-34, A@-33, C@-32, A@-31.
minus_35	:-	T@-36, T@-35, G@-34, A@-33, C@-32.
minus_10	:-	T@-14, A@-13, T@-12, A@-11, A@-10, T@-9.
minus_10	:-	T@-13, A@-12, A@-10, T@-8.
minus_10	:-	T@-13, A@-12, T@-11, A@-10, A@-9, T@-8.
minus_10	:-	T@-12, A@-11, T@-7.
conformation	:-	C@-47, A@-46, A@-45, T@-43, T@-42, A@-40, C@-39, G@-22, T@-18, C@-16, G@-8, C@-7, G@-6, C@-5, C@-4, C@-2, C@-1.
conformation	:-	A@-45, A@-44, A@-41.
conformation	:-	A@-49, T@-44, T@-27, A@-22, T@-18, T@-16, G@-15, A@-1.
conformation	:-	A@-45, A@-41, T@-28, T@-27, T@-23, A@-21, A@-20, T@-17, T@-15, T@-4.

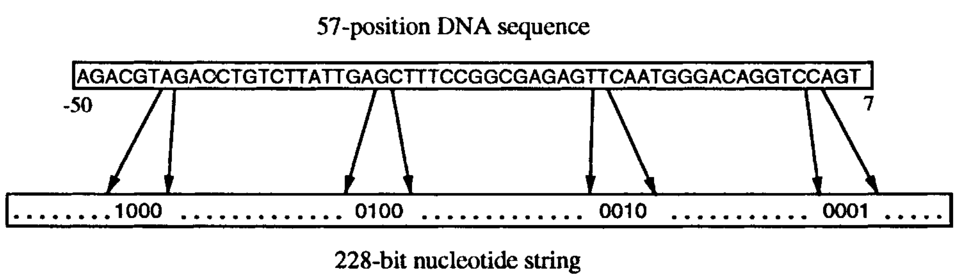

Fig. 8. A 57-position DNA sequence. Each position takes one of the four nucleotide values {A, G, T, C}. Using local representation, each DNA sequence is expanded into a 228-bit nucleotide string. This version of 106-case promoter data set, obtained from the UCI machine learning database repository, contains no missing value.

TABLE V
PERFORMANCE OF FUZZY ARTMAP, CASCADE ARTMAP, AND CASCADE ARTMAP RULES ON THE PROMOTER DATA SET COMPARING WITH THE SYMBOLIC LEARNING ALGORITHM ID-3, THE KNN SYSTEM, CONSENSUS SEQUENCE ANALYSIS, THE BACKPROPAGATION NETWORK, THE KBANN SYSTEM, AND THE NOFM RULES

Systems	# Nodes/Rules	# Antecedent	Error (%)
ID-3	-	-	17.9
KNN (K=3)	105	-	12.3
Consensus Sequences	-	-	11.3
Backpropagation Network	16	-	7.5
Fuzzy ARTMAP	20.6	-	6.5
KBANN	16	-	2.9
Cascade ARTMAP	13+15.9	-	2.0
NOFM rules	12	100	3.8
Cascade ARTMAP rules	19.5	53.1	3.0

dropped as a condition for promoters, so are the four rules defining it. All the minus_35 and minus_10 rules are preserved, but have been refined to refer to only two salient nucleotide bases. Two new rules for identifying promoters are created, which contain features of *minus_35* and *conformation*. These two rules are believed to compensate for the elimination of *conformation*. Eight nonpromoter rules are created. They are slightly more irregular due to the randomness of nonpromoters.

The confidence factor attached to each ARTMAP rule provides another dimension for interpreting the rule. By having a confidence factor of one, the first promoter rule is very frequently used and thus important. It is activated by different combinations of minus_35 and minus_10 rules, each individually does not have a high usage. The two new promoter rules are roughly of equal importance but are not as heavily used as the first promoter rule. The first three minus_35 rules are

TABLE VI
A SET OF PROMOTER RULES EXTRACTED FROM CASCADE ARTMAP. THE SET CONSISTS OF 19 RULES AND A TOTAL OF 46 ANTECEDENTS. THE REAL NUMBER ASSOCIATED WITH EACH RULE REPRESENTS THE RULE'S CONFIDENCE FACTOR

promoter (1.00)	**:- minus_35, minus_10.**
promoter (0.31)	**:- A@-45, G@-34.**
promoter (0.22)	**:- G@-34, T@-25, T@-18.**
minus_35 (0.41)	**:- G@-34, C@-32.**
minus_35 (0.34)	**:- T@-36, T@-35.**
minus_35 (0.22)	**:- A@-33, C@-32.**
minus_35 (0.03)	**:- T@-36, C@-32.**
minus_10 (0.44)	**:- A@-12, T@-8.**
minus_10 (0.31)	**:- A@-13, T@-9.**
minus_10 (0.19)	**:- A@-11, T@-7.**
minus_10 (0.06)	**:- A@-9, T@-8.**
non-promoter (0.19)	**:- A@5.**
non-promoter (0.16)	**:- A@-49, C@6, G@.7**
non-promoter (0.16)	**:- A@7.**
non-promoter (0.16)	**:- T@-23.**
non-promoter (0.12)	**:- A@-15, T@1.**
non-promoter (0.12)	**:- C@-46, G@-26.**
non-promoter (0.06)	**:- T@-34, T@-33, C@-27, T@-26, G@5.**
non-promoter (0.03)	**:- A@-45, T@-44, G@-42, T@-29, A@-24, T@-7, A@6, G@7.**

TABLE VII
A SET OF PROMOTER RULES EXTRACTED BY THE NOFM ALGORITHM FROM KBANN. IT CONSISTS OF NINE RULES AND A TOTAL OF 83 ANTECEDENTS. A COMPRESSED NUCLEOTIDE REPRESENTATION REFERS TO BASES BY STARTING A SEQUENCE LOCATION FOLLOWED BY A SUBSEQUENCE. FOR EXAMPLE, @-37 'C-T-' INDICATES C@-37 AND T@-35. THE FUNCTION "NT()" RETURNS THE NUMBER OF ENCLOSED ANTECEDENTS THAT MATCH AN INPUT SEQUENCE. STANDARD NUCLEOTIDE AMBIGUITY CODES ARE INTERPRETED AS FOLLOWS: M=A/C, K=G/T, R=A/G, D=A/G/T, W=A/T, B=C/G/T, AND S=C/G

promoter	**:-**	**minus_35, minus_10.**		
minus_10	**:-**	**10**	**<**	**3.8 * nt (@-14 '-TA-A-T-') + 3.0 * nt (@-14 '-G–C—') - 3.0 * nt (@-14 '-A–T—').**
minus_10	**:-**	**2**	**of**	**@-14 '–CA—T' and not 1 of @-14 '–RB—S'.**
minus_10	**:-**	**10**	**<**	**3.0 * nt (@-14 '-TAT–T-') + 1.8 * nt (@-14 '—GA–').**
minus_10	**:-**	**1**	**<**	**3.5 * nt (@-14 'TAWAAY–') - 1.7 * nt (@-14 '-T–TG–') - 2.2 * nt (@-14 'CSSK-A–').**
minus_35	**:-**	**10**	**<**	**4.0 * nt (@-37 '-TTGAT–') + 1.5 * nt (@-37 '—TCC–') - 1.5 * nt (@-37 '-RGAGG–').**
minus_35	**:-**	**10**	**<**	**5.1 * nt (@-37 '-T-G–A-') + 3.1 * nt (@-37 '-GT——') - 1.9 * nt (@-37 '-CGW—-') - 3.1 * nt (@-37 '-A—-C-').**
minus_35	**:-**	**3**	**of**	**@-37 'C-TGAC–'.**
minus_35	**:-**			**@-37 '-TTG-CA-'.**

more highly utilized than the last minus_35 rule. A similar pattern is observed for the minus_10 rules. The nonpromoter rules have lower and less contrasting confidence values. The first four nonpromoter rules nevertheless seem slightly more important. The last two nonpromoter rules have the least confidences, and could be dropped with little degradation of overall performance.

Table VII shows a set of promoter rules extracted by the NofM algorithm from KBANN [19]. The NofM rule set consists of only nine rules but contains 83 countable antecedents.

Moreover, the rules make use of several complex constructs, including NofM, a counting function "nt," addition, subtraction, multiplication, and comparison of real numbers. Also, the NofM rules involve seven nucleotide ambiguity codes, and have already employed a compressed format for representing adjacent nucleotide bases to simplify rules. Comparing complexity, ARTMAP rules are much cleaner and easier to interpret. More importantly, by preserving the symbolic rule form during learning, the extracted rules are identical in form and can be compared directly with the original rules. Furthermore, the use of confidence factors enables ranking of rules. This is particularly important to human experts in analyzing the rules.

ACKNOWLEDGMENT

This article is based on a chapter of a doctoral dissertation submitted to Boston University. The author wishes to acknowledge his dissertation committee members, G. A. Carpenter, S. Grossberg, and M. Cohen, for their guidance and support. Thanks also go to the three anonymous referees who provided valuable comments and suggestions.

REFERENCES

[1] Y. R. Asfour, G. A. Carpenter, S. Grossberg, and G. W. Lesher, " Fusion ARTMAP: A neural-network architecture for multichannel data fusion and classification," in *Proc. World Congr. Neural Networks*, Portland, OR, vol. II. Hillsdale, NJ: Lawrence Erlbaum, July 1993, pp. 210–215.

[2] G. A. Carpenter and S. Grossberg, " A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graphics, Image Processing*, vol. 37, pp. 54–115, 1987.

[3] 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.

[4] G. A. Carpenter, S. Grossberg, and J. H. Reynolds, "ARTMAP: Supervised real time learning and classification by a self-organizing neural network," *Neural Networks*, vol. 4, pp. 565–588, 1991.

[5] 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.

[6] G. A. Carpenter and A.-H. Tan, " Fuzzy ARTMAP: Rule extraction, and medical databases," in *Proc. World Congress on Neural Networks*, vol. I. Hillsdale, NJ: Lawrence Erlbaum, July 1993, pp. 501–506.

[7] ——, "Rule extraction: From neural architecture to symbolic representation," *Connection Sci.*, vol. 7, no. 1, pp. 3–27, 1995.

[8] R. Farber, A. Lapedes, and K. Sirotkin, " Determination of eucaryotic protein coding regions using neural networks and information theory," *J. Molecular Biol.*, vol. 226, pp. 471–479, 1992.

[9] L. M. Fu and L. C. Fu, " Mapping rule-based knowledge into neural architecture," *Knowledge-Based Syst.*, vol. 3, pp. 48–56, 1990.

[10] R. C. Lacher, S. I. Hruska, and D. C. Kuncicky, " Backpropagation learning in expert networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 62–72, 1992.

[11] A. Lapedes, C. Barnes, C. Burks, R. Farber, and K. Sirotkin, " Application of neural networks and other machine learning algorithms to DNA sequence analysis," in *Computers and DNA, SFI Studies in the Sciences of complexity*, vol. VII. Reading, MA: Addison-Wesley, 1990, pp. 265–281.

[12] P. M. Murphy and D. W. Aha, " UCI repository of machine learning databases [machine-readable data repository]," Univ. California, Irvine, Dep. Inform. Comput. Sci., 1992.

[13] M. O'Neill, " Escherichia coli promoters, vol I: Consensus as it relates to spacing class, specificity, repeat substructure, and three-dimensional organization," *J. Biol. Chem.*, vol. 264, pp. 5522–5530, 1989.

[14] ——, "Escherichia coli promoters, vol. II: A spacing class-dependent promoter search protocol," *J. Biol. Chem.*, vol. 264, pp. 5531–5534, 1989.

[15] J. R. Quinlan, " Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[16] A.-H. Tan, " Synthesizing neural network and symbolic knowledge processing," Ph.D. dissertation, Dep. Cognitive Neural Syst., Boston Univ., MA, Apr. 1994.

[17] ——, "Integrating rules and neural computation," in *Proc. 1995 IEEE Int. Conf. Neural Networks*, Perth, Australia, vol. IV, 1995, pp. 1794–1799.

[18] G. G. Towell and J. W. Shavlik, " Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules," in *Advances in Neural Inform. Processing Syst. 4*. San Mateo, CA: Morgan Kaufmann, 1992, pp. 977–984.

[19] ——, "Extracting refined rules from knowledge-based neural networks," *Machine Learning*, vol. 13, pp. 71–101, 1993.

[20] G. G. Towell, J. W. Shavlik, and M. O. Noordewier, "Refinement of approximately correct domain theories by knowledge-based neural networks," in *Proc., 8th Nat. Conf. AI*, Boston, MA: AAAI Press, 1990, pp. 861–866.

[21] V. Tresp, J. Hollatz, and S. Ahmad, " Network structuring and training using rule-based knowledge," in *Advances in Neural Inform. Processing Syst. 5*. San Mateo, CA: Morgan Kaufmann, 1993, pp. 977–984.

[22] M. S. Waterman, *Mathematical Methods for DNA Sequences* . Boca Raton, FL: CRC, 1989.

[23] M. S. Waterman and L. Gordon, " Multiple hypothesis testing for sequence comparisons," in *Computers and DNA, SFI Studies in the Sciences of Complexity*, vol. VII. Reading, MA: Addison-Wesley, 1990, pp. 127–135.

[24] P. H. Winston, *Artificial Intelligence*, 3rd ed. Reading, MA: Addison-Wesley, 1992.

[25] L. Zadeh, "Fuzzy Sets," *Inform. Contr.*, PP. 338–353, 1965.

Ah-Hwee Tan received the B.Sc. degree with honors in 1989 and the M.Sc. degree in 1991, both in computer and information science, from the National University of Singapore. He received the Ph.D. degree in cognitive and neural systems from Boston University, MA, in 1994.

He is an Associate Member of the research staff of the Institute of Systems Science at the National University of Singapore. He has published over 25 technical papers on neural networks and knowledge-based systems. His research interests include integration of neural-network and symbolic processing, cognitive modeling, agent architecture, computational learning theory, pattern recognition, and associative memory.

Dr. Tan is a member of the International Neural Network Society and the Singapore Computer Society.