1-1995

Rule extraction: From neural architecture to symbolic representation

Gail A. CARPENTER

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Rule Extraction: From Neural Architecture to Symbolic Representation

GAIL A. CARPENTER & AH-HWEE TAN

(Received for publication 1 February 1994; revised paper accepted August 1994)

This paper shows how knowledge, in the form of fuzzy rules, can be derived from a supervised learning neural network called fuzzy ARTMAP. Rule extraction proceeds in two stages: pruning, which simplifies the network structure by removing excessive recognition categories and weights; and quantization of continuous learned weights, which allows the final system state to be translated into a usable set of descriptive rules. Three benchmark studies illustrate the rule extraction methods: (1) Pima Indian diabetes diagnosis, (2) mushroom classification and (3) DNA promoter recognition. Fuzzy ARTMAP and ART-EMAP are compared with the ADAP algorithm, the k *nearest neighbor system, the back-propagation network and the* C4.5 *decision tree. The ARTMAP rule extraction procedure is also compared with the Knowledgetron and* NOFM *algorithms, which extract rules from back-propagation networks. Simulation results consistently indicate that ARTMAP rule extraction produces compact sets of comprehensible rules for which accuracy and complexity compare favorably to rules extracted by alternative algorithms.*

KEYWORDS: Fuzzy ARTMAP, rule, confidence factor, pruning.

1. Introduction: Rules and Fuzzy ARTMAP

Fuzzy ARTMAP (Carpenter *et al.*, 1992) is a neural network architecture that performs incremental supervised learning of recognition categories (pattern classes) and multi-dimensional maps for both binary and analog input patterns. When performing classification tasks, fuzzy ARTMAP formulates recognition categories of input patterns, and associates each category with its respective prediction. The knowledge that ARTMAP discovers during learning is equivalent to if–then rules which link sets of antecedents to their consequents. At any point during incremental learning, the system architecture can be translated into a set of rules.

G.A. Carpenter and A-H. Tan, Center for Adaptive Systems and Department of Cognitive and Neural Systems, Boston University, 111 Cummington Street, Boston, MA 02215, USA. E-mail: gail@cns.bu.edu and ahhwee@iss.nus.sg. This work was supported in part by ARPA (ONR-N00014-92-J-4015), the National Science Foundation (NSF-IRI-94-01659) and the Office of Naval Research (ONR-N00014-91-J-4100) (GAC) and by a graduate fellowship from the Institute of Systems Science, National University of Singapore (A-HT).

0954-0091/95/010003-25

However, this set of rules may be too large to be readily comprehensible. This paper describes a procedure to reduce a trained fuzzy ARTMAP system to a compact rule-based representation that maintains the predictive accuracy of the full network, and compares the complexity and performance of this system to that of other neural network, machine learning and rule-extraction algorithms.

Rules can be derived more readily from an ARTMAP network than from a back-propagation network, in which the role of hidden units is usually not explicit. In fuzzy ARTMAP, each category node in the F_2^a field (Figure 1) roughly corresponds to a rule. Each node has an associated weight vector that can be directly translated into a verbal or algorithmic description of the antecedents in the corresponding rule. However, large databases typically cause ARTMAP to generate too many rules to be of practical use. The goal of the rule-extraction task is thus to select a small set of highly predictive category nodes and to describe them in a comprehensible form. To evaluate a category node, a confidence factor that measures both usage and accuracy is computed. Removal of low-confidence recognition categories created by atypical examples produces smaller networks. Removal of redundant weights in a category node's weight vector reduces the number of antecedents in the corresponding rule. Further, in order to describe the knowledge in a simplified rule form, real-valued weights are quantized into a small set of values.

The ARTMAP rule-extraction algorithm has been evaluated using the Pima Indian diabetes (PID) data set obtained from the UCI repository of machine learning databases (Murphy & Aha, 1992). Simulation results (Carpenter & Tan, 1993) show that pruning consistently produces rule sets that are more accurate

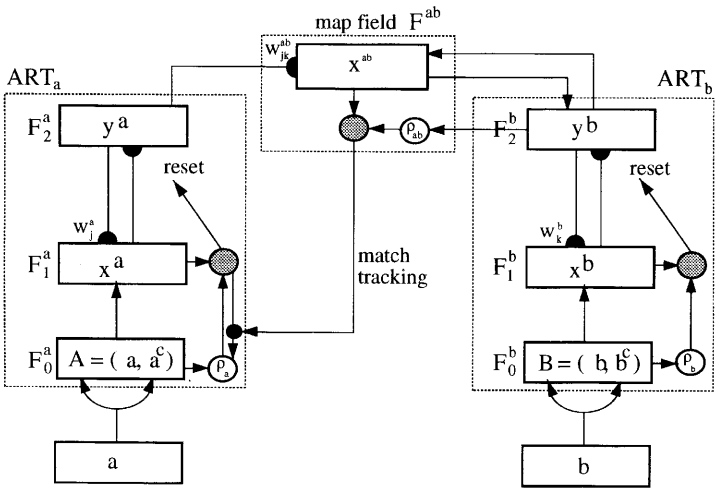

Figure 1. Fuzzy ARTMAP architecture (Carpenter *et al.*, 1992). The ART_a complement coding preprocessor transforms the M_a-vector **a** into the $2M_a$-vector $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$ at the ART_a field F_0^a. **A** is the input vector to the ART_a field F_1^a. Similarly, the input 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 **b** or to a previously uncommitted ART_a category node.

than the full system but only one-third the size. Quantization produces more comprehensible rules at only a slight cost in terms of performance.

In addition to the PID benchmark, this paper reports two other benchmark studies that compare ARTMAP rule extraction with two other neural network rule-extraction algorithms. Performance was assessed in terms of predictive accuracy and system complexity. Predictive accuracy was measured by the performance of the extracted rules on a test set that was unseen during both training and rule extraction. The number of rules and antecedents in the rule set determine system complexity.

The mushroom data set (Schlimmer, 1987) benchmark problem partitions sensory feature vectors into two classes, edible or poisonous. ARTMAP rule extraction is compared with a back-propagation rule-extraction system called Knowledgetron (Fu, 1992). Simulation results indicate that the ARTMAP pruning procedure derives rules that are more accurate and far fewer in number than the Knowledgetron conjunctive rules.

The final data set, which recognizes promoters in DNA sequences, is an expanded version of the promoter data set maintained by Craven and Shavlik (1993, 1994). Fuzzy ARTMAP and ART-EMAP are compared with the K nearest neighbour (KNN) algorithm, the back-propagation network and the *C4.5* decision tree. The ARTMAP rule-extraction procedure is also compared with the NofM algorithm that extracts rules from back-propagation networks (Craven & Shavlik, 1993, 1994). Preliminary simulations (Tan, 1994) indicated that while the performance of ARTMAP rules was equivalent to that of NofM rules, ARTMAP had faster learning but NofM had better code compression. In this paper, a new ARTMAP rule pruning strategy and an antecedent pruning procedure further reduce ARTMAP complexity. Using the revised pruning method on the DNA promoter simulations, the ARTMAP rule sets are comparable to NofM rules both in accuracy and in system size, while maintaining the fast learning advantage.

To make this article self-contained, Section 2 provides a summary of fuzzy ART and ARTMAP. Section 3 describes the ARTMAP rule-extraction algorithm. Section 4 reviews the Knowledgetron, NofM, ADAP, KNN and *C4.5* algorithms. The final section reports simulation results of the PID, mushroom and DNA promoter benchmark studies.

2. Fuzzy ARTMAP

ARTMAP (Carpenter *et al.*, 1991a) is a neural network for supervised learning and prediction of binary inputs. Fuzzy ARTMAP (Carpenter *et al.*, 1992) generalizes ARTMAP to classify inputs by a fuzzy set of features, or a pattern of fuzzy membership values that indicate the extent to which each feature is consistently present or absent with respect to a given category. This generalization is accomplished by replacing the ART 1 modules (Carpenter & Grossberg, 1987, 1991) of binary ARTMAP with fuzzy ART modules (Carpenter *et al.*, 1991b). Where ART 1 dynamics are described in terms of set-theoretic operations, fuzzy ART dynamics are described in terms of fuzzy set-theoretic operations (Zadeh, 1965; Kosko, 1986) (Figure 2).

Each ARTMAP system includes a pair of adaptive resonance theory modules (ART_a and ART_b) that create stable recognition categories in response to arbitrary sequences of input patterns (Figure 1). During supervised learning, ART_a receives

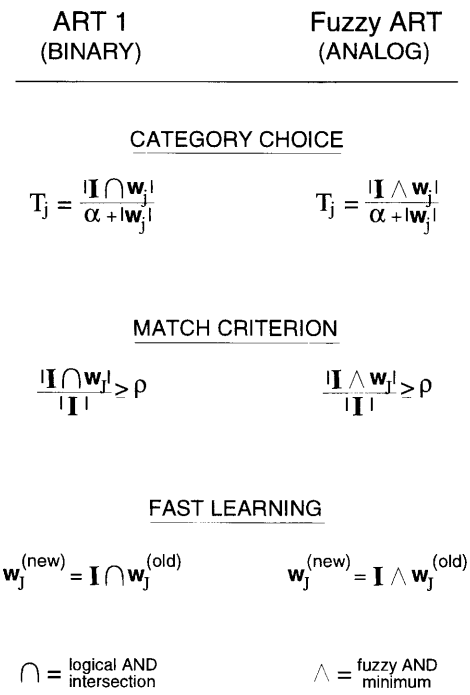

Figure 2. Analogy between ART 1 and fuzzy ART.

a stream $\{\mathbf{a}^{(p)}\}$ of input patterns and ART_b receives a stream $\{\mathbf{b}^{(p)}\}$ of input patterns, where $\mathbf{b}^{(p)}$ is the correct prediction given $\mathbf{a}^{(p)}$. An associative learning network and an internal controller link these modules to make the ARTMAP system operate in real time. The controller creates the minimal number of ART_a recognition categories, or 'hidden units', needed to meet accuracy criteria. A minimax learning rule enables ARTMAP to learn quickly, efficiently and accurately as it conjointly minimizes predictive error and maximizes code compression. This scheme automatically links predictive success to category size on a trial-by-trial basis using only local operations. It works by increasing the ART_a vigilance parameter (ρ_a) by the minimal amount needed to correct a predictive error at ART_b.

An ART_a baseline vigilance parameter $\bar{\rho}_a$ calibrates the minimum confidence needed for ART_a to accept a chosen category, rather than search for a better one through automatically controlled search. Lower values of $\bar{\rho}_a$ enable larger categories to form, maximizing code compression. Initially, $\rho_a = \bar{\rho}_a$. During training, a predictive failure at ART_b increases ρ_a by the minimum amount needed to trigger ART_a search, through a feedback control mechanism called match tracking. Match tracking sacrifices the minimum amount of compression necessary to correct the predictive error. Hypothesis testing selects a new ART_a category, which focuses attention on a cluster of $\mathbf{a}^{(p)}$ input features that is better able to predict $\mathbf{b}^{(p)}$. With fast learning, match tracking allows a single ARTMAP system to learn a different prediction for a rare event than for a cloud of similar frequent events in which it is embedded.

2.1. Fuzzy ART

Fuzzy ART incorporates computations from fuzzy set theory into ART systems. The crisp (non-fuzzy) intersection operator ($\cap$) that describes ART 1 dynamics is replaced by the fuzzy AND operator ($\wedge$) of fuzzy set theory in the choice, search and learning laws of ART 1 (Figure 2). By replacing the crisp logical operations of ART 1 with their fuzzy counterparts, fuzzy ART can learn stable categories in response to either analog or binary patterns.

2.1.1. ART field activity vectors. Each ART system includes a field F_0 of nodes that represent a current input vector, a field F_2 that represents the active code, or category and a field F_1 that receives both bottom-up input from F_0 and top-down input from F_2. Vector $\mathbf{I} = (I_1, \ldots, I_M)$ denotes F_0 activity, with each component I_i in the interval [0,1], for $i = 1, \ldots, M$. Vector $\mathbf{x} = (x_1, \ldots, x_M)$ denotes F_1 activity and $\mathbf{y} = (y_1, \ldots, y_N)$ denotes F_2 activity. The number of nodes in each field can be arbitrarily large.

2.1.2. Weight vector. Associated with each F_2 category node j ($j = 1, \ldots, N$) is a vector $\mathbf{w}_j \equiv (w_{j1}, \ldots, w_{jM})$ 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. Each component w_{ji} can decrease but never increase during learning. Thus, each weight vector $\mathbf{w}_j(t)$ converges to a limit. The fuzzy ART weight, or prototype, vector $\mathbf{w}_j$ subsumes both the bottom-up and top-down weight vectors of ART 1.

2.1.3. 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.

2.1.4. 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 $\wedge$ (Zadeh, 1965) 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 $\mathcal{J}$ denotes the chosen category, where

$$T_{\mathcal{J}} = \max\{T_j : j = 1, \ldots, N\} \tag{5}$$

If more than one T_j is maximal, the category with the smallest j index is chosen. In particular, nodes become committed in order $j = 1, 2, 3, \ldots$. When the $\mathcal{J}$th category is chosen, $y_{\mathcal{J}} = 1$; and $y_j = 0$ for $j \neq \mathcal{J}$. In a choice system, the F_1 activity

vector $\mathbf{x}$ obeys the equation

$$\mathbf{x} = \begin{cases} \mathbf{I} & \text{if } F_2 \text{ is inactive} \\ \mathbf{I} \wedge \mathbf{w}_{\mathcal{J}} & \text{if the } \mathcal{J} \text{ th } F_2 \text{ node is chosen} \end{cases} \tag{6}$$

2.1.5. Resonance or reset. Resonance occurs if the match function $|\mathbf{I} \wedge \mathbf{w}_{\mathcal{J}}|/|\mathbf{I}|$ of the chosen category meets the vigilance criterion

$$\frac{|\mathbf{I} \wedge \mathbf{w}_{\mathcal{J}}|}{|\mathbf{I}|} \geq \rho \tag{7}$$

that is, by (6), when the $\mathcal{J}$th category becomes active, resonance occurs if

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_{\mathcal{J}}| \geq \rho |\mathbf{I}| \tag{8}$$

Learning then ensues, as defined below. Mismatch reset occurs if

$$\frac{|\mathbf{I} \wedge \mathbf{w}_{\mathcal{J}}|}{|\mathbf{I}|} < \rho \tag{9}$$

that is, if

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_{\mathcal{J}}| < \rho |\mathbf{I}| \tag{10}$$

Then the value of the choice function $T_{\mathcal{J}}$ is set to 0 for the duration of the input presentation to prevent the persistent selection of the same category during search. A new index $\mathcal{J}$ represents the active category, selected by (5). The search process continues until the chosen $\mathcal{J}$ satisfies the matching criterion (7).

2.1.6. Learning. Once search ends, the weight vector $\mathbf{w}_{\mathcal{J}}$ learns according to the equation

$$\mathbf{w}_{\mathcal{J}}^{(\text{new})} = \beta(\mathbf{I} \wedge \mathbf{w}_{\mathcal{J}}^{(\text{old})}) + (1 - \beta)\mathbf{w}_{\mathcal{J}}^{(\text{old})} \tag{11}$$

Fast learning corresponds to setting $\beta = 1$. The learning law of the NGE system (Salzberg, 1990) is equivalent to (11) in the fast-learn limit with the complement coding.

2.1.7. Fast-commit, slow-recode. For efficient coding of noisy input sets, it is useful to set $\beta = 1$ when $\mathcal{J}$ is an uncommitted node, and then to take $\beta < 1$ for slower adaptation after the category is already committed. The fast-commit, slow-recode option makes $\mathbf{w}_{\mathcal{J}}^{(\text{new})} = \mathbf{I}$ the first time category $\mathcal{J}$ becomes active. Moore (1989) introduced the learning law (11), with fast commitment and slow recoding, to investigate a variety of generalized ART 1 models. Some of these models are similar to fuzzy ART, but none uses complement coding. Moore describes a category proliferation problem that can occur in some analog ART systems when many random inputs erode the norm of weight vectors. Complement coding solves this problem.

2.1.8. Normalization by complement coding. Normalization of fuzzy ART inputs prevents category proliferation. The $F_0 \rightarrow F_1$ inputs are normalized if, for some $\gamma > 0$,

$$\sum_i I_i = |\mathbf{I}| \equiv \gamma \tag{12}$$

for all inputs $\mathbf{I}$. One way to normalize each vector $\mathbf{a}$ is

$$\mathbf{I} = \frac{\mathbf{a}}{|\mathbf{a}|} \tag{13}$$

Complement coding normalizes the input but it also preserves amplitude information, in contrast to (13). Complement coding represents both the on-response and the off-response to an input vector **a** (Figure 1). In its simplest form, **a** represents the on-response and $\mathbf{a}^c$, the complement of **a**, represents the off-response, where

$$a_i^c \equiv 1 - a_i \tag{14}$$

The complement coded $F_0 \rightarrow F_1$ input **I** is the $2M$-dimensional vector

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, \ldots, a_M, a_1^c, \ldots, a_M^c) \tag{15}$$

A complement coded input is automatically normalized, because

$$\begin{aligned} |\mathbf{I}| &= |(\mathbf{a}, \mathbf{a}^c)| \\ &= \sum_{i=1}^{M} a_i + \left(M - \sum_{i=1}^{M} a_i \right) \\ &= M \end{aligned} \tag{16}$$

With complement coding, the initial condition

$$w_{j\,1}(0) = \cdots = w_{j,2M}(0) = 1 \tag{17}$$

replaces the fuzzy ART initial condition (1).

The close linkage between fuzzy subsethood and ART choice/search/learning forms the foundation of the computational properties of fuzzy ART. In the conservative limit, where the choice parameter $\alpha = 0^+$, the choice function $T_j\mathbf{I}(2)$ measures the degree to which $\mathbf{w}_j$ is a fuzzy subset of **I** (Kosko, 1986). A category $\mathcal{J}$ for which $\mathbf{w}_\mathcal{J}$ is a fuzzy subset of **I** will then be selected first, if such a category exists. Resonance depends on the degree to which **I** is a fuzzy subset of $\mathbf{w}_\mathcal{J}$, by (7) and (9). When $\mathcal{J}$ is such a fuzzy subset choice, then the match function value is

$$\frac{|\mathbf{I} \wedge \mathbf{w}_\mathcal{J}|}{|\mathbf{I}|} = \frac{|\mathbf{w}_\mathcal{J}|}{|\mathbf{I}|} \tag{18}$$

Choosing $\mathcal{J}$ to maximize $|\mathbf{w}_j|$ among fuzzy subset choices, by (2), thus maximizes the opportunity for resonance in (7). If reset occurs for the node that maximizes $|\mathbf{w}_\mathcal{J}|$, then reset will also occur for all other subset choices.

2.2. ARTMAP Prediction and Search

Fuzzy ARTMAP incorporates two fuzzy ART modules ART_a and ART_b that are linked together via an inter-ART module F^{ab} called a map field. The map field forms predictive associations between categories and realizes the ARTMAP match tracking rule. Match tracking increases the ART_a vigilance parameter ρ_a in response to a predictive error, or mismatch, at ART_b. Match tracking reorganizes category structure so that subsequent presentations of the input do not repeat the error. An outline of the ARTMAP algorithm follows.

2.2.1. 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, $\mathbf{I} = \mathbf{B} = (\mathbf{b}, \mathbf{b}^c)$ (Figure 1). Variables in ART_a or ART_b are designated by subscripts or superscripts a or b. For ART_a, $\mathbf{x}^a \equiv (x_1^a \ldots x_{2M_a}^a)$ denotes the F_1^a output vector; $\mathbf{y}^a \equiv (y_1^a \ldots y_{N_a}^a)$ denotes the F_2^a output vector; and $\mathbf{w}_j^a \equiv (w_{j\,1}^a, w_{j\,2}^a, \ldots, w_{j,2M_a}^a)$ denotes the jth ART_a weight vector. For ART_b, $\mathbf{x}^b \equiv (x_1^b \ldots x_{2M_b}^b)$ denotes the F_1^b output vector; $\mathbf{y}^b \equiv (y_1^b \ldots y_{N_b}^b)$ denotes

the F_2^b output vector; and $\mathbf{w}_k^b \equiv (w_{k1}^b, w_{k2}^b, \ldots, w_{k,2M_b}^b)$ denotes the kth ART$_b$ weight vector. For the map field, $\mathbf{x}^{ab} \equiv (x_1^{ab}, \ldots, x_{N_b}^{ab})$ denotes the F^{ab} output vector, and $\mathbf{w}_j^{ab} \equiv (w_{j1}^{ab}, \ldots, w_{jN_b}^{ab})$ denotes the weight vector from the jth F_2^a node to F^{ab}. Vectors $\mathbf{x}^a$, $\mathbf{y}^a$, $\mathbf{x}^b$, $\mathbf{y}^b$, and $\mathbf{x}^{ab}$ are reset to $\mathbf{0}$ between input presentations.

2.2.2. ART$_b$ for classification. When the output vector $\mathbf{b}$ represents a binary classification, such as poisonous or edible mushrooms, ART$_b$ structure and function become simplified. Then, both F_1^b nodes and F_2^b nodes correspond to the set of output classes. An ART$_b$ bottom-up/top-down match is either perfect, when a predicted class is the same as the supervised learning vector $\mathbf{b}$; or a complete mismatch, when the two differ. Thus, all ART$_b$ vigilance values between 0 and 1 give equivalent performance. All simulations in this article are classification problems.

2.2.3. 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 $\mathcal{J}$ sends input to the map field F^{ab} via the weights $\mathbf{w}_{\mathcal{J}}^{ab}$. An active F_2^b node K sends input to F^{ab} via one-to-one pathways between F_2^b and F^{ab}. Thus, for classification examples, F^{ab} nodes also represent the set of possible predictive classes. 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 the following:

$$\mathbf{x}^{ab} = \begin{cases} \mathbf{y}^b \wedge \mathbf{w}_{\mathcal{J}}^{ab}, & \text{if the } \mathcal{J}\text{th } F_2^a \text{ node is active and } F_2^b \text{ is active} \\ \mathbf{w}_{\mathcal{J}}^{ab}, & \text{if the } \mathcal{J}\text{th } F_2^a \text{ node is active and } F_2^b \text{ is inactive} \\ \mathbf{y}^b, & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is active} \\ \mathbf{0}, & \text{if } F_2^a \text{ is inactive and } F_2^b \text{ is inactive} \end{cases} \tag{19}$$

By (19), $\mathbf{x}^{ab} = \mathbf{0}$ if $\mathbf{y}^b$ fails to confirm the map field prediction made by $\mathbf{w}_{\mathcal{J}}^{ab}$. Such a mismatch event triggers an ART$_a$ search for a better category, as follows.

2.2.4. Match tracking. At the start of each input presentation, ART$_a$ vigilance ρ_a equals a baseline vigilance parameter $\bar{\rho}_a$. 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

$$|\mathbf{x}^{ab}| < \rho_{ab}|\mathbf{y}^b| \tag{20}$$

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 $|\mathbf{A} \wedge \mathbf{w}_{\mathcal{J}}^a|\,|\mathbf{A}|^{-1}$. Then, since

$$|\mathbf{x}^a| = |\mathbf{A} \wedge \mathbf{w}_{\mathcal{J}}^a| < \rho_a|\mathbf{A}| \tag{21}$$

ART$_a$ fails to meet the matching criterion, as in (10), and the search for another F_2^a node begins. The search leads to a different chosen F_2^a node $\mathcal{J}$ with

$$|\mathbf{x}^a| = |\mathbf{A} \wedge \mathbf{w}_{\mathcal{J}}^a| \geq \rho_a|\mathbf{A}| \tag{22}$$

and

$$|\mathbf{x}^{ab}| = |\mathbf{y}^b \wedge \mathbf{w}_{\mathcal{J}}^{ab}| \geq \rho_{ab}|\mathbf{y}^b| \tag{23}$$

If no such node exists and if all F_2^a nodes are already committed, F_2^a automatically shuts down for the remainder of the input presentation.

2.2.5. Map field learning. Weights w_{jk}^{ab} in $F_2^a \rightarrow F^{ab}$ paths initially satisfy

$$w_{jk}^{ab}(0) = 1 \tag{24}$$

During resonance with the ART_a category J active, $\mathbf{w}_J^{ab}$ approaches the map field vector $\mathbf{x}^{ab}$ as in (11). With fast learning, once J learns to predict the ART_b category K, that association is permanent, i.e. $w_{JK}^{ab} = 1$ and $w_{Jk}^{ab} = 0$ $(k \neq K)$ for all time.

2.2.6. Voting. Fast learning implies that category structure depends significantly on the order of input presentation. When a given training set is presented to two ARTMAP networks with different input orderings, the two ART_a category structures, and the corresponding input–output predictions, may differ, even if the accuracy of the two networks is the same. Overall predictive accuracy can benefit from this order dependence through a simple voting strategy, as follows. Each of several voting ARTMAP networks is trained on a given input set presented in random order. For the test set input, the number of networks predicting a given outcome is counted. The prediction receiving the most votes wins. Voting tends to improve performance because the strategy helps to cancel errors of a given system that occur near a noisy region or decision boundary. Moreover, the number of votes provides a measure of certainty for a given prediction.

3. ARTMAP Rule Extraction

In an ARTMAP network, 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 with 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 (Figure 3). The number of rules equals the number of committed F_2^a nodes.

3.1. Rule Pruning

To reduce the complexity of fuzzy ARTMAP, a rule-pruning procedure aims to select a small set of rules from a trained network, based on each rule's confidence

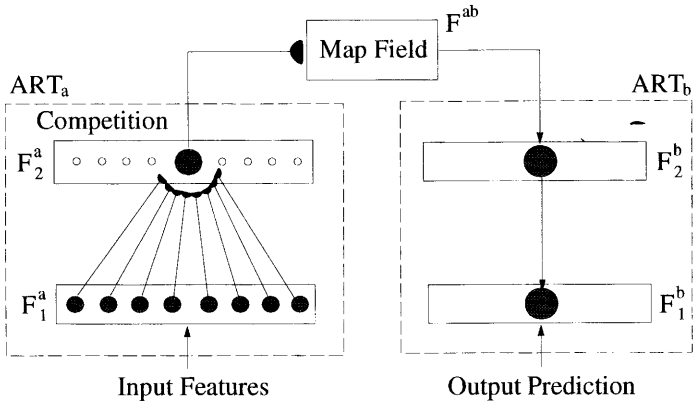

Figure 3. Schematic diagram of a rule in fuzzy ARTMAP. Each F_2^a node maps a prototype feature vector (antecedents) to a prediction (consequent).

factor. To derive concise rules, an antecedent pruning procedure aims to remove antecedents from rules while preserving accuracy. The rule-extraction process partitions the input set into a training set, a predicting set and a test set. The training set provides inputs for ARTMAP weight adaptation. The predicting set provides an accuracy estimate for each rule in the full network. The test set then measures performance of the reduced network, following rule extraction. In order to ensure fair comparison of algorithms in benchmark simulations, a fixed test set is normally used to evaluate final performance of all algorithms. If an algorithm, such as back-propagation, uses only a training and a test set, that training set is split into training and predicting subsets for ARTMAP learning and rule extraction.

The rule-pruning algorithm derives a confidence factor for each F_2^a category node in terms of its usage frequency in the training set and its predictive accuracy on the predicting set. The confidence factor identifies good rules with nodes that are frequently used and generally correct, as well as nodes that are rarely used but highly accurate. ARTMAP removes rules that have low confidence. Overall performance is actually improved when the pruning algorithm removes rules that represent misleading special cases from the training set.

Specifically, the pruning algorithm evaluates an F_2^a recognition category j in terms of a confidence factor CF_j:

$$CF_j = (1-\gamma)U_j + \gamma A_j \tag{25}$$

where U_j is the usage of node j, A_j is its accuracy, and $\gamma \in [0,1]$ is a weighting factor. Typically, setting $\gamma = 0.6$ would allow a rare ($U_j \cong 0$) but accurate ($A_j \cong 1$) rule j to have a confidence factor ($CF_j \cong 0.6$) that would survive a pruning threshold in the usual range of 0.5–0.6.

3.1.1. Usage. For an ART_a category j that predicts outcome k, its usage U_j is defined to be the number of training set patterns coded by any node j (C_j), divided by the maximum number ($C_\mathcal{J}$) of training patterns coded by any node $\mathcal{J}$ that predicts k:

$$U_j = C_j / \max\{C_\mathcal{J}\text{: node } \mathcal{J} \text{ predicts outcome } k\} \tag{26}$$

Usage is thus scaled to lie between 0 and 1, with U_j equal to 1 for at least one node j among those predicting outcome k.

3.1.2. Accuracy. For an ART_a category j that predicts outcome k its accuracy A_j is defined to be the per cent (P_j) of predicting set patterns predicted correctly by node j, divided by the maximum per cent ($P_\mathcal{J}$) of patterns predicted correctly by any node $\mathcal{J}$ that predicts outcome k:

$$A_j = P_j / \max\{P_\mathcal{J}\text{: node } \mathcal{J} \text{ predicts outcome } k\} \tag{27}$$

Accuracy is thus also scaled to lie between 0 and 1, with A_j equal to 1 for at least one node j among those predicting each outcome k. Scaling within each outcome class k ensures that at least one F_2^a node in the pruned network will predict each outcome.

After confidence factors are determined, recognition categories are removed from the network using a threshold pruning strategy or a local pruning strategy, as follows.

3.1.3. 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 an initial elimination of nodes that are infrequently used and inaccurate. To avoid over-pruning, it is sometimes useful to specify a minimum number of recognition categories to be preserved in the system.

3.1.4. 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. In simulations, this option, called hill-climbing, gives slightly larger rule sets but better predictive accuracy. A hybrid strategy first applies threshold pruning and then applies local pruning to the remaining smaller set of rules.

3.2. Antecedent Pruning

During rule extraction, a non-zero 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 makes a predictive error, each antecedent of the rule that also appears in the current input has its error factor increased in proportion to the smaller of its magnitudes in the rule and in the input vector. After the error factor for each antecedent is determined, a local pruning strategy, similar to the one for rules, removes redundant antecedents by setting the corresponding weight equal to zero.

3.3. Quantizing Weight Values

With analog input patterns or slow learning, ARTMAP learns real-valued weights. In order to describe the rules in words rather than real numbers, the feature values represented by weights w_{ji}^a are quantized. A quantization level Q is defined as the number of feature values in the quantized rules. For example, with $Q = 3$, feature values are described as low, medium or high in the fuzzy rules. Quantization by truncation divides [0,1] into Q intervals and assigns a quantization point to the lower bound of each interval, i.e. $V_q = (q-1)/Q$ for $q = 1, \ldots, Q$. When a weight falls in interval q, the algorithm reduces its value to V_q. Quantization by round-off distributes Q quantization points evenly, with one at each end point, i.e. $V_q = (q-1)/(Q-1)$. The algorithm then rounds off a weight to the nearest V_q value.

4. Comparative Algorithms

Knowledgetron and NOFM are algorithms that extract symbolic rules from back-propagation networks. Each uses a clustering technique during training to facilitate rule extraction. Simulations in Section 5 compare ARTMAP performance with these and with benchmark simulations of the ADAP, KNN and *C4.5* systems. For reference, these comparison algorithms are briefly described here.

4.1. Knowledgetron

The Knowledgetron algorithm (Fu, 1992) consists of the Knowledgetron back-

propagation (KTBP) trainer and the Knowledgetron (KT) translator. The KTBP trainer iterates a process of adapting a multi-layer neural network and clustering hidden units to encode information more compactly. The KT translator searches the rule space for confirming and disconfirming rules. Positive (negative) attributes refer to attributes which link to a unit with positive (negative) weights. For each unit, the algorithm forms confirming rules by exploring combinations of positive attributes and negated negative attributes that activate the unit. Similarly, the algorithm forms disconfirming rules by exploring combinations by negative attributes and negated positive attributes.

4.2. NofM Algorithm

The NofM algorithm (Table I) constructs rules of the form:

If N of the M antecedents are true, then ...

from a trained feedforward network. The NofM algorithm was originally used to extract symbolic rules from knowledge-based neural networks in which the toplogy and initial weights had been specified by an approximately correct domain theory (Towell & Shavlik, 1992). The NofM algorithm was extended by Craven and Shavlik (1993, 1994) who trained back-propagation networks using soft weight-sharing (Nowlan & Hinton, 1992), which encourages weights to form clusters during training.

4.3. C4.5

On a DNA promoter data set, Craven and Shavlik showed that the NofM algorithm induced rules with better predictive accuracy than rules produced by the symbolic learning algorithm *C4.5* (Quinlan, 1993). The *C4.5* algorithm constructs a decision tree based on attribute values. Its pruning and rule-generation processes made it a natural choice for the comparative studies of Craven and Shavlik (1994).

4.4. ADAP

The ADAP (adaptive) learning routine (Smith *et al.*, 1988) has previously been applied to the PID data set (Section 5.1). The three-layer ADAP architecture (Figure 4) uses fixed connections from the sensory layer to the association layer, and error feedback to adapt connections from the association layer to the responder layer.

4.5. KNN

The KNN algorithm is a look-up system that stores all training patterns in its memory. The category of a test pattern is determined by a vote of the categories of

Table I. A summary of the NofM algorithm

(1) *Clustering*: The weights converging on each hidden and output unit are grouped into clusters using the *join* algorithm (Hartigan, 1975), a standard clustering method.
(2) *Averaging*: The value of each weight is set to the average value of the weights in its cluster.
(3) *Eliminating*: Clusters that are not needed to activate correctly a unit are eliminated.
(4) *Optimizing*: Unit biases are retrained after the changes.
(5) *Extracting*: Each hidden and output unit is translated into a set of N-of-M rules that describe the conditions under which the unit will be activated.

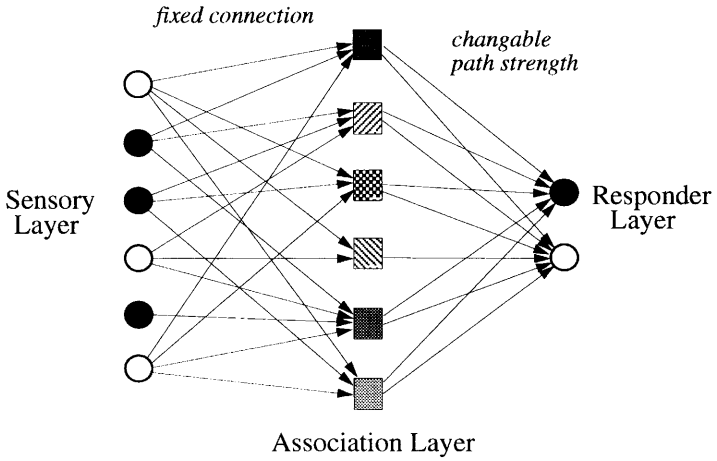

Figure 4. The ADAP architecture (Smith *et al.*, 1988).

the K training patterns closest to the test pattern. KNN is often most useful when the available training set is small. However, both memory requirements and search time scale up rapidly for large data sets.

5. Comparative Simulations

5.1. PID Diagnosis (cf: ADAP)

The PID task seeks to determine whether a patient develops diabetes, based on eight clinical indices. The PID data set (Table II) contains 768 cases, of which the 268 positive cases (35%) are from patients diagnosed as diabetic. Smith *et al.* (1988) converted the 8 feature values into 37 binary variables. The simulations used 100 000 association units (p. 263). Real-valued predictions were converted to binary predictions using a cut-off of 0.448. After training on 576 inputs, the sensitivity (per cent correct of actual positive cases) and specificity (per cent correct of actual negative cases) on the 192 test set cases were each 76%.

In order to compare fuzzy ARTMAP with ADAP performance, each simulation used the same 192 test set inputs. For extracting ARTMAP rules, the remaining 576 cases were partitioned into a training set of 384 inputs and a predicting set of 192 inputs. During pruning, the weighting factor γ and the pruning threshold τ were each fixed at 0.5. Two indices are used to compare test set performance, namely (1) accuracy, equal to the per cent correct by binary prediction, and (2) the c-index. The c-index is an evaluation of the predictive score that is independent of the relative frequency of positive and negative outcomes. It is equal to the average probability, over all possible pairs of cases with different outcomes, that the classifier will assign a higher score to a positive case. To apply a voting strategy, an ARTMAP system was trained in 20 simulation runs with inputs presented in different orderings. For each test case, voting produced a net predictive score between 0 and 1 for each outcome, equal to the fraction of times that network predicts that outcome. The entire simulation, including the training of fuzzy ARTMAP, extraction of rules and performance evaluation, was repeated ten times.

Table II. The eight input features of a PID input and their statistics. The PID data set was obtained from the UCI machine learning repository (Murphy & Aha, 1992)

No.	Feature	Description	Mean	SD
1	PREG	Number of times pregnant	3.8	3.4
2	PGC	Plasma glucose concentration	120.9	32.0
3	DBP	Diastolic blood pressure (mm Hg)	69.1	19.4
4	TSFT	Triceps skin fold thickness (mm)	20.5	16.0
5	SI	2-hour serum insulin (μU ml^{-1})	79.8	115.2
6	BMI	Body mass index	32.0	7.9
7	DPF	Diabetes pedigree function	0.5	0.3
8	AGE	Age (years)	33.2	11.8

Table III. PID simulation results of ADAP, fuzzy ARTMAP and systems obtained by combinations of rule extraction methods. ARTMAP results are obtained by voting across 20 simulations. Rule pruning improves test set performance, while quantization gradually degrades performance

			Training		Predicting		Testing	
Methods	Data set partition	No. nodes/rules	Accuracy	c-index	Accuracy	c-index	Accuracy	c-index
ADAP	576/192	100000	—	—	—	—	76.0	—
Fuzzy ARTMAP	576/192	63.5 (49–82)	100.0	1.000	—	—	75.9	0.819
Threshold pruning	384/192/192	19.6 (12–30)	86.7	0.940	88.2	0.942	78.5	0.848
Pruning + $Q = 10$	384/192/192	19.7 (11–28)	79.3	0.854	82.7	0.855	**79.0**	0.842
Pruning + $Q = 5$	384/192/192	19.6 (11–28)	75.7	0.801	80.7	0.804	77.5	0.829
Pruning + $Q = 3$	384/192/192	19.6 (12–26)	70.5	0.737	69.9	0.725	69.3	0.731

Voting fuzzy ARTMAP is about as accurate as ADAP but uses far fewer nodes (Table III). With fast learning, ARTMAP correctly learned all 576 training predictions after 6 to 15 input presentations (epochs). The reduced rule sets do not classify correctly all the training patterns, but actually show better performance on the test set (Table III). In particular, the threshold pruning procedure yields about one-third as many rules but gives better test-set performance as measured by both accuracy and the c-index. Quantization degrades performance gradually as the number of quantized steps Q decreases. Quantization with $Q = 3$ uses simpler rules, with 'low, medium and high' as feature values, but produces significantly poorer performance. A good compromise uses $Q = 5$ quantized steps.

Table IV shows six PID rules extracted by rule pruning and quantization ($Q = 5$). Each row can be directly translated into a fuzzy rule. Because of complement coding, fuzzy ARTMAP learns a pair of weights for each feature. These weights specify a minimum and a maximum value, or interval, for each feature in each rule. For example, row 1 can be interpreted as the rule shown in Table V. The rules extracted can be evaluated and adapted by medical experts. Novel rules discovered through the rule-extraction process may provide new insights for human and machine diagnosis.

Table IV. Six PID rules extracted by pruning and quantization ($Q = 5$). The pruned set of 23 rules predicts correctly 78.2% training, 77.6% predicting and 76.0% test set vectors. Each rule is described in terms of a range of quantized feature values. Interpretation of weight values: 1 = very low, 2 = low, 3 = medium, 4 = high and 5 = very high. '1–5' means a feature is irrelevant

	Feature weights								Rule statistics			Test set	
Predict	PREG	PGC	DBP	TSFT	SI	BMI	DPF	AGE	Usage	Accuracy	CF	No.	Accuracy
+	3–5	3–5	3–5	1–3	1–2	1–4	1–2	2–4	1.00	0.80	0.90	7	0.71
+	1–2	3–5	3–4	1–3	1–4	3–4	1–3	1–2	0.78	0.62	0.70	18	0.83
+	1–3	3–5	3–5	1–3	1–5	3	1–3	1–4	0.33	1.00	0.67	8	0.88
–	1–2	3–4	3–4	1–2	1–2	2–3	1–2	1–2	1.00	0.94	0.97	19	0.89
–	1–2	2–4	3–4	1–3	1–2	2–4	1–3	1–2	0.54	0.88	0.71	12	0.92
–	1–2	3–4	3–4	1–3	1–2	2–3	1–2	1–2	0.38	1.00	0.69	10	1.00

Table V. Interpretation of rule 1 in the sample PID rule set (Table IV)

If number of times pregnant is medium to very high
  and plasma glucose concentration is medium to very high
  and diastolic blood pressure is medium to very high
  and triceps skin fold thickness is very low to medium
  and 2-hour serum insulin is below medium
  and body mass index is not very high
  and diabetes pedigree function is below medium
  and age is not extreme
THEN diabetes is likely

5.2. Mushroom Classification (cf: Knowledgetron)

The mushroom classification problem is to determine whether a mushroom is edible or poisonous based on its observable features. The mushroom database (Schlimmer, 1987) consists of 8124 instances, each of which is characterized by 22 sensory features (Table VI). Inputs that represent 3916 poisonous mushrooms constitute 48.2% of the total population.

On this problem, Fu (1992) used 1000 inputs to train a back-propagation network containing 127 input units, 63 hidden units, 2 output units and 8127 connections. The network classified the 1000 training cases with 100% accuracy and a disjoint test set of 1000 cases with 99.0% accuracy. Knowledgetron then generated a system of 233 rules, which classified the 1000 training cases with 100% accuracy and the 1000 test cases with 99.6% accuracy.

In ARTMAP simulations, the 22 input features were converted into 125 binary attributes. Complement coding was applied to represent both the presence and absence of each attribute. ARTMAP learning and testing were performed with parameter values $\alpha = 0.001$, $\beta = 1$ and $\bar{\rho}_a = 0$. The simulation results averaged over 20 runs are summarized in Table VII. When ARTMAP is trained with 1000 cases, an average of 5.8 rules are created, compared to the 233 of Knowledgetron. Simulations are 99.8% accurate over the remaining 7124 cases.

In the rule-extraction simulations, 1000 cases were used as the training set, 1000 cases as the predicting set, and the system was tested on the remaining 6124

Table VI. The 22 input features of the mushroom data set (Schlimmer, 1987)

No.	Features	Values
1	cap-shape	bell/conical/convex/flat/knobbed/sunken
2	cap-surface	fibrous/grooves/scaly/smooth
3	cap-color	brown/buff/cinnamon/gray/green/pink/purple/red/white/yellow
4	bruises	true/false
5	odor	almond/anise/creosote/fishy/foul/musty/none/pungent/spicy
6	gill-attachment	attached/descending/free/notched
7	gill-spacing	close/crowded/distant
8	gill-size	broad/narrow
9	gill-color	black/brown/buff/chocolate/gray/green/orange/pink/purple/ red/white/yellow
10	stalk-shape	enlarging/tapering
11	stalk-root	bulbous/club/cup/equal/rhizomorphs/rooted/missing
12	stalk-surface-above-ring	fibrous/scaly/silky/smooth
13	stalk-surface-below-ring	fibrous/scaly/silky/smooth
14	stalk-color-above-ring	brown/buff/cinnamon/gray/orange/pink/red/white/yellow
15	stalk-color-below-ring	brown/buff/cinnamon/gray/orange/pink/red/white/yellow
16	veil-type	partial/universal
17	veil-color	brown/orange/white/yellow
18	ring-number	none/one/two
19	ring-type	cobwebby/evanescent/flaring/large/none/pendant/sheathing/zone
20	spore-print-color	black/brown/buff/chocolate/green/orange/purple/white/yellow
21	population	abundant/clustered/numerous/scattered/several/solitary
22	habitat	grasses/leaves/meadows/paths/urban/waste/woods

Table VII. Comparison between ARTMAP, ARTMAP rule-extraction methods, back-propagation and Knowledgetron on the mushroom data set. Data partitions indicate the number of training, predicting and test patterns; number of rules shows the average number and range of rules; number of antecedents counts the total number of antecedents summing over all rules

Systems	Data partition	No. rules	No. antecedents	Train (%)	Predict (%)	Test (%)
Back-propagation	1000/0/1000	—	—	100.0	—	99.0
Knowledgetron	1000/0/1000	233	—	100.0	—	99.6
ARTMAP	1000/0/7124	5.8 (4–7)	—	100.0	—	99.8
Rule pruning	1000/1000/6124	5.1 (4–7)	366.0	99.9	99.9	99.8
Antecedent pruning	1000/1000/6124	5.1 (4–7)	51.0	99.9	99.9	99.8

cases. Rule pruning and antecedent pruning used the local pruning strategy (Section 3.1) with $\gamma = 0.5$ in the confidence factor equation (25). As shown in Table VII, the rule and antecedent pruning procedures combine to remove an average of 315 antecedents but only 0.7 rules from the already small sets of ARTMAP rules. After pruning, systems with an average of 5.1 rules and 51 antecedents maintain predictive accuracy at 99.8%. Table VIII shows a sample set of four ARTMAP rules created in the simulations.

Table VIII. A sample set of four ARTMAP rules with a total of 22 antecedents for classifying mushrooms. These rules classify correctly 99.9% of the 8124 mushrooms in the data set

Predict	Edible (Conf 1.00 Usage 1.00 Accuracy 1.00 Predict 835 Test Acc 1.00)
IF	ring-type is not none
and	spore-print-color is not chocolate
Predict	Edible (Conf 0.85 Usage 0.69 Accuracy 1.00 Predict 2286 Test Acc 1.00)
IF	cap-surface is not grooves
and	odor is not creosote, not foul, and not pungent
and	gill-color is not buff
and	stalk-surface-below-ring is not scaly
and	spore-print-color is not green
and	population is not abundant and not numerous
and	habitat is not meadows
Predict	Poisonous (Conf 0.78 Usage 0.56 Accuracy 1.00 Predict 483 Test Acc 1.00)
IF	habitat is not waste
Predict	Poisonous (Conf 1.00 Usage 1.00 Accuracy 1.00 Predict 2520 Test Acc 1.00)
IF	odor is not almond, and not anise
and	stalk-color-above-ring is not gray
and	stalk-color-below-ring is not gray
and	veil-color is not brown, and not orange
and	population is not abundant, not numerous, and not solitary

5.3. DNA Promoter Recognition (cf: KNN, Back-propagation, NoFM, C4.5 and ART-EMAP)

The third simulation is that of recognizing promoters in DNA sequences. Promoters are short nucleotide sequences that occur before genes and serve as binding sites for the enzyme RNA polymerase during gene transcription. The promoter data set (Craven & Shavlik, 1993) is an expanded version of the 106-case promoter data set in the UCI repository. It consists of 468 patterns, half of which are positive instances (promoters). Each input pattern represents a 57-position window, with the leftmost 50 window positions labeled -50 to -1 and the rightmost seven labeled 1 to 7 (Figure 5). Each position is a nominal feature which takes one of the four nucleotide values (A, G, T, C) or unknown (?). Using local representation, each DNA sequence is expanded into a 228-bit (57×4) nucleotide string.

In the following sections, fuzzy ARTMAP and ARTMAP rule-extraction algorithms are compared with KNN, *C4.5*, back-propagation and NoFM algorithms. Note that each system handles missing values differently. KNN replaces missing values by ones; a back-propagation network, and hence NoFM, assigns 0.25 to missing features; ARTMAP replaces them with zeros; and *C4.5* distributes missing values probabilistically according to relative frequencies of known cases. Since only about 1% of feature values are missing, however, these variations are of minor importance in the present simulation.

5.3.1. KNN simulations. Test set performance of KNN on the promoter data set was very accurate (Figure 6), with a minimum error rate of 5.5% obtained using $K = 30$ neighbors. However, since KNN performs no data compression, it is most useful for problems with small data sets. For the promoter data set, KNN stores all 468 patterns with a total of 26676 attributes. The NoFM and ARTMAP rule-

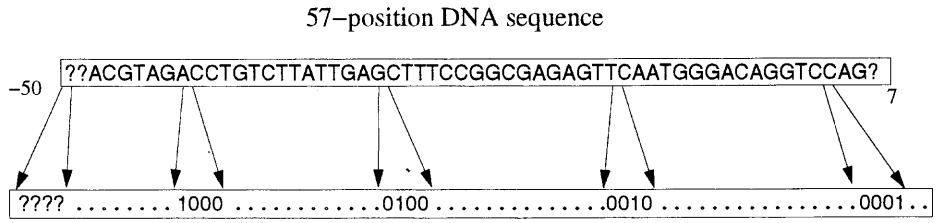

Figure 5. 57-position DNA sequence. Each position takes one of the four nucleotide values (A, G, T, C) or unknown (?). Using local representation, each DNA sequence is expanded into a 228-bit nucleotide string.

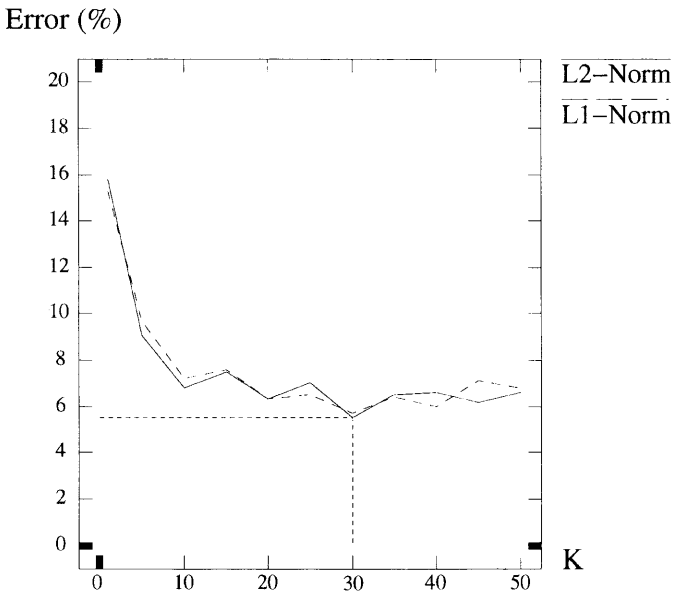

Figure 6. Average predictive error rate of KNN on the promoter data set over 100 runs using $K = 1$ to 50 neighbors.

extraction simulations create approximately 10 to 20 rules with about 100 antecedents, but have error rates around 10%, making a trade-off between system complexity and predictive accuracy.

5.3.2. Back-propagation, NofM and C4.5 simulations. Craven and Shavlik (1993, 1994) used the promoter data set to evaluate their NofM algorithm against the symbolic learning system *C4.5* (Quinlan, 1993). Using a 10-fold cross-validation methodology, the accuracy and the system size of back-propagation networks and NofM rules were compared with those of a *C4.5* decision tree and extracted *C4.5* rules.

A back-propagation network trained using soft weight sharing achieved an error rate of 7.9% (Table IX), less than half the 16.9% error rate of *C4.5* decision trees. The NofM rules produced an error rate of 11.1%, still lower than the 13.5% error of *C4.5* rules. However, the *C4.5* rules were more concise than the NofM

Table IX. Performance of *C4.5* decision trees, *C4.5* rules, back-propagation networks with soft-weight sharing and NOFM rules on the promoter data set

Systems	No. rules	No. antecedents	Error (%)
C4.5 decision trees	—	—	16.9
C4.5 rules	23.2	47.3	13.5
Back-propagation	—	—	7.9
NOFM rules	8.2	119.6	11.1
NOFM rules (after pruning)	8.1	97.2	10.2

rules in terms of the number of antecedents. To reduce the system size, Craven and Shavlik derived a rule/antecedent pruning algorithm which reduced both the complexity and the error rate of NOFM rules.

5.3.3. ART-EMAP spatial evidence accumulation. The promoter data set has very few (468) examples, given the dimension of its input vectors (228). For such problems with sparse data points, the ART-EMAP spatial evidence accumulation process (Carpenter & Ross, 1993), which integrates distributed activity across F_2^a category nodes, is effective for classifying noisy or novel inputs. In ARTMAP systems with category choice, only the F_2^a node $\mathcal{J}$ that receives maximal $F_1^a \to F_2^a$ input T_j^a predicts ART_b output. In simulations with category choice

$$y_j^a = \begin{cases} 1, & \text{if } j = \mathcal{J} \text{ where } T_{\mathcal{J}}^a > T_\lambda^a \text{ for all } \lambda \neq \mathcal{J} \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

as in (5). ART-EMAP uses the choice rule (28) during the initial period of supervised learning. However, during performance, the F_2^a output vector $\mathbf{y}^a$ represents a less extreme contrast enhancement of the $F_1^a \to F_2^a$ input $\mathbf{T}^a$. Two algorithms that approximate contrast enhancement by competitive networks (Grossberg, 1973) are studied below.

Power rule. The power rule, as used in the ART-EMAP system, raises T_j^a to a power p and normalizes the total activity:

$$y_j^a = \frac{(T_j^a)^p}{\sum_\lambda (T_\lambda^a)^p} \tag{29}$$

(Figure 7(a)). The power rule converges toward the choice rule as p becomes large.

K-max rule. In the spirit of the KNN system, the K-max rule makes a prediction based on the set of K F_2^a nodes receiving the largest $F_1^a \to F_2^a$ input T_j^a. The F_2^a activities y_j^a are then

$$y_j^a = \begin{cases} \dfrac{T_j^a}{\sum_{\lambda \in \Lambda} T_\lambda^a}, & \text{if } j \in \Lambda \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

where Λ is the set of K category nodes with the largest T_j^a values (Figure 7(b)). The K-max rule with $K = N$ is equivalent to the power rule with $p = 1$; and the K-max rule with $K = 1$ is equivalent to the choice rule (28).

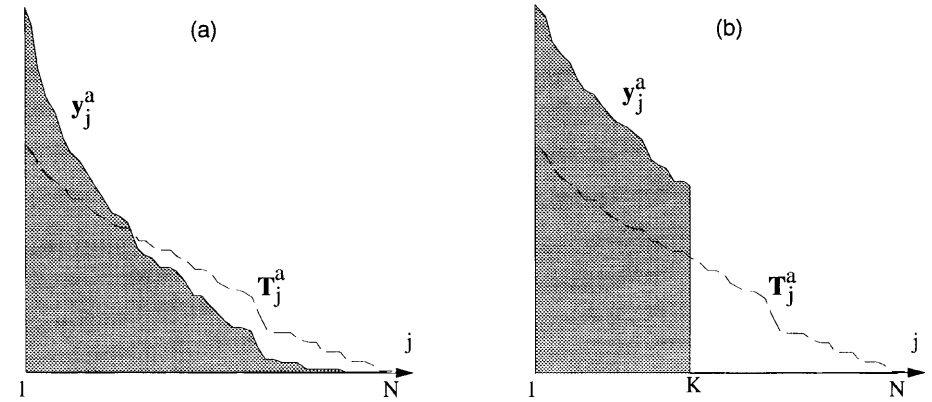

Figure 7. (a) Contrast enhancement by the power rule with $p = 2$. (b) Contrast enhancement by the K-max rule. T_j^a is the input to the j th F_2^a node and y_j^a is the contrast enhanced activity of node j.

After the F_2^a activity vector $\mathbf{y}^a$ is contrast enhanced by (29) or (30), the input vector $\mathbf{x}^{ab}$ from F_2^a to the map field F^{ab} is

$$\mathbf{x}^{ab} = \sum_j \mathbf{w}_j^{ab} y_j^a \tag{31}$$

(Figure 1).

5.3.4. ARTMAP simulations. The 10-fold cross-validation methodology of Craven and Shavlik (1993, 1994) was also used to evaluate ARTMAP performance. The data set was divided into ten partitions. ARTMAP was trained on nine partitions and tested on the remaining partition, using parameter values $\alpha = 10$, $\beta = 1$ and $\bar{\rho}_a = 0$. The real-valued scores produced by ART-EMAP spatial evidence accumulation were thresholded at 0.5 to produce a binary prediction.

The ART-EMAP power rule and the K-max rule both perform consistently better than the choice rule (Figure 8). F_2^a choice is equivalent to the K-max rule when $K = 1$ and to the power rule as $p \to \infty$. When $K = N$, the K-max rule is the same as the power rule with $p = 1$. The K-max rule reduces the error rate from 17.8% with $K = 1$ (choice) to 8.0% with $K = 20$ predictive categories. The power rule performs best with distributed F_2^a activity $\mathbf{y}^a$. The best performance (7.4% error) is obtained with $p = 2$ (Figures 7(a) and 8). Both the power rule and the K-max rule simulations indicate that distributed F_2^a activity improves predictive accuracy compared to compressed code representations. All rule-extraction simulations use the power rule with $p = 2$.

5.3.5. ARTMAP rule extraction. In ARTMAP rule-extraction simulations of the DNA promoter data set, seven of the ten input partitions were used to train and evaluate usage U_j of each rule; two partitions were used to evaluate accuracy A_j; and the remaining one partition was used to test the extracted rules. Setting the usage/accuracy weighting factor γ equal to 0.6 in (25) gave 60% weight to accuracy and 40% to usage in the confidence factor. To avoid over-pruning, the system always preserved a minimum of 36 rules. No quantization was needed, since the binary input patterns and fast learning produce binary weights.

For distributed ART-EMAP prediction, a slightly different definition of usage U_j aims to preserve a more representative mix of rules for each outcome in the

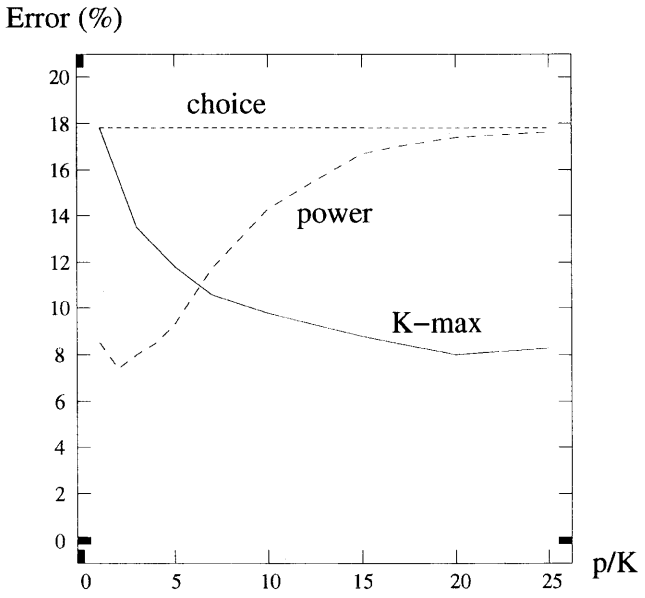

Figure 8. Ten-run average predictive error rate of ART-EMAP on the promoter data set, compared to fuzzy ARTMAP choice at F_2^a. Parameter p is the power used in the power rule (29) and K is the number of F_2^a recognition categories used in the K-max rule (30). Compression decreases from choice to a linear representation of the input as p decreases from ∞ to 1 for the power rule or as K increases from 1 to N for the K-max rule.

pruned set. Recall that in (26), for a node j that predicts outcome k, C_j is the number of training set patterns coded by node j. Usage is here defined to be

$$U_j = F_j / \max_{\mathcal{J}}\{F_{\mathcal{J}}\} \tag{32}$$

where F_j is the fraction of training set patterns with outcome k that are coded by node j:

$$F_j = C_j / (\#\text{ training set patterns with outcome } k) \tag{33}$$

Table X summarizes ARTMAP performance on the promoter data set together with those obtained by back-propagation, NOFM and KNN algorithms. The performance of fuzzy ARTMAP is slightly better than that of the back-propagation network. A pruning threshold $\tau = 0.6$ reduces the number of ARTMAP rules from 117 to 39.3, which increases the error rate from 7.4% to 9.8%. Local rule pruning with hill-climbing takes away an additional 13.7 rules and 98.5 antecedents. Finally, local antecedent pruning removes over half of the antecedents from the remaining rules. The resultant rule sets, with an average of 19.9 rules and 87.5 antecedents, produce a predictive error of 10.4%, which is comparable to that of the NOFM rules. Comparing system complexity, NOFM has fewer rules but more antecedents than ARTMAP.

Also reported in Table X are the results of 10-voting ARTMAP. Under the voting strategy, an ARTMAP system is trained in multiple simulation runs using different orderings of a fixed input set. The output predictions of ARTMAP across runs are averaged to form a final prediction for each test case. This technique was

Table X. Performance of ARTMAP networks and pruning methods on the promoter data set compared to back-propagation networks, the NOFM algorithm and the KNN system. ARTMAP networks used the ART-EMAP power rule with $p = 2$

Systems	No. rules	No. antecedents	Error (%)
KNN ($K = 30$)	468	26676	5.5
Back-propagation	—	—	7.9
NOFM rules	8.2	119.6	11.1
NOFM rules (after pruning)	8.1	97.2	10.2
Fuzzy ARTMAP	117.0	—	7.4
Threshold rule pruning	39.3	286.6	9.8
Local rule pruning	25.6	188.1	10.3
Local antecedent pruning	19.9	87.5	10.4
Voting ARTMAP	(voting across 10 simulations)		5.5
Threshold rule pruning	(voting across 10 simulations)		7.0

used by Towell *et al.* (1990) in the promoter simulations to obtain a slight improvement in performance, but the rule-extraction simulations of Craven and Shavlik (1993, 1994) did not utilize this technique. When extracting rules, the predictions of rule sets extracted across runs are averaged to form a final prediction. Voting gives ARTMAP a significant improvement in performance. Even after threshold pruning, the rules still perform slightly better (7.0% error) than the full network (7.4% error).

5.3.6. Semantic interpretation and comparison. Table XI shows a sample set of rules extracted from a fuzzy ARTMAP system. Rules with consequents $P_1, P_2, \ldots, P_6$ are created by positive instances (promoters) and rules with consequents $N_1, N_2, \ldots, N_{10}$ are created by negative instances (non-promoters). The system prediction is made by summing evidence for promoters across the positive rules. Note that the positive rules for identifying promoters in Table XI are quite simple, while the negative rules for identifying non-promoters are slightly more complicated. This is perhaps due to greater variations among non-promoters. Certain interesting regularities in the rules can also be observed. For example, features like T@-36, T@-35 and G@-34 consistently appear across the positive rules but not in the negative rules. This suggests that these features are good indicators for promoters.

The form of ARTMAP rules differs from the form of NOFM rules (Table XII). ARTMAP creates rules for predicting both promoters and non-promoters. NOFM focuses only on positive instances (promoters), then uses a two-choice assumption to identify non-promoters. By using ART-EMAP evidence accumulation, ARTMAP rules include intermediate variables ($P_1, P_2, \ldots, P_6$ and $N_1, N_2, \ldots, N_{10}$) that correspond to F_2^a category nodes, whereas the intermediate variables of NOFM rules correspond to hidden units in back-propagation networks. The negative weights in back-propagation networks allow NOFM rules to include negative terms; in ARTMAP rules, the effect of negative terms could be included through complement coding of input patterns. Complement coding, however, was not used in ARTMAP promoter simulations, to keep the input dimension smaller.

Neither system requires an exact match to fire a rule. NOFM rule firing is based on the satisfaction of individual conditions (NOFM) stated within each rule. ARTMAP rule firing is based on competition among all positive and negative

Table XI. A sample set of 17 ARTMAP rules consisting of 68 antecedents. The antecedent notation T@-36 indicates the nucleotide *T* in the position 36 nucleotides before the start of a putative gene

Promoter	Feature template/conditions	Rule statistics		
		CF	Usage	Accuracy
	:- $\sum_{i=1}^{6} f(P_i) > 0.5$ where $f(x) = x^2/(\sum_{i=1}^{6} P_i^2 + \sum_{i=1}^{10} N_i^2)$			
P_1	:- T@-36 G@-34 T@-13 A@-12 T@-8	1.00	1.00	1.00
P_2	:- T@-36 G@-34 T@-30 A@-11	0.92	0.80	1.00
P_3	:- T@-35 T@-14 A@-13 T@-12 A@-10	0.88	0.70	1.00
P_4	:- G@-37 T@-35 A@-31	0.84	0.60	1.00
P_5	:- T@-36	0.84	0.60	1.00
P_6	:- T@-38	0.60	0.50	0.67
N_1	:- C@-13 C@-6	0.80	0.50	1.00
N_2	:- A@-17 G@6	0.76	0.40	1.00
N_3	:- C@-15 T@-14 G@-1	0.76	0.40	1.00
N_4	:- G@-33 G@-15 C@-5	0.76	0.40	1.00
N_5	:- C@-43 G@-26 G@-24 G@-21	0.76	0.40	1.00
N_6	:- G@-47 G@-37 T@-34 A@-28 C@-12 G@-11 G@-3 C@4 C@5	0.72	0.30	1.00
N_7	:- C@-35 A@-34 C@-10 G@-7 G@3	0.72	0.30	1.00
N_8	:- C@-40 G@-2 G@7	0.72	0.30	1.00
N_9	:- A@-8 G@-6	0.64	0.10	1.00
N_{10}	:- G@-4 T@7	0.50	0.50	0.50

Table XII. Sample NofM rules extracted from a back-propagation promoter network (Craven & Shavlik, 1994)

promoter	:- 2 of	{hidden-3, hidden-4, hidden-5}.
hidden-3	:- 7 of	{not (A@-36), not (G@-35), not (A@-34), not (G@-33), C@-32, not (C@-31), not (C@-21), not (C@-15), T@-12, T@-8}.
hidden-4	:- 10 of	{not (G@-44), not (C@-36), T@-35, not (G@-33), not (G@-32), not (C@-31), not (G@-13), not (C@-12), A@-11, not (G@-10), not (G@-9), not (G@-8), T@-7, not (G@2)}.
hidden-5	:- 4 of	{T@-36, not (A@-35), not (G@-13), A@-10, not (G@-3)}.

rules. Considering each rule independently, an ARTMAP positive or negative rule is roughly equivalent to a $|\mathbf{w}_j^a|$-of-*M* rule. However, when functioning as a whole, each ARTMAP rule affects the others' activities through spatial evidence accumulation. Another feature of ARTMAP rules is that each carries a confidence factor which reflects its rate of use and reliability.

6. Conclusion

ARTMAP rules and NofM rules are roughly comparable in terms of both predictive accuracy and system complexity. Since NofM rules are derived from a back-propagation network, training must be off-line, with slow learning, while ARTMAP networks are designed to maintain stability even with fast, incremental training. Simulations in this paper also illustrate how ARTMAP networks, with

rule extraction, have better code compression than alternative algorithms such as ADAP, Knowledgetron and KNN, while maintaining or improving predictive accuracy.

Acknowledgements

We wish to thank Carol Jefferson and Robin Locke for their valuable assistance in the preparation of the manuscript. We also wish to thank Mark Craven and Jude Shavlik for sharing the promoter data set and NOFM simulation details. Finally, we are grateful to David Touretzky for suggesting the DNA promoter benchmark and for providing useful comments on the comparison between NOFM and ARTMAP rules.

References

Carpenter, G.A. & Grossberg, S. (1987) A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, **37**, 54–115.

Carpenter, G.A. & Grossberg, S. (Eds) (1991) *Pattern Recognition by Self-Organizing Neural Networks*. Cambridge, MA: MIT Press.

Carpenter, G.A. & Ross, W.D. (1993) ART-EMAP: a neural network architecture for object recognition by evidence accumulation. In *World Congress on Neural Networks*, Portland, OR, Vol. III. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 649–656.

Carpenter, G.A. & Tan, A.H. (1993) Fuzzy ARTMAP, rule extraction and medical databases. In *World Congress on Neural Networks*, Portland, OR, Vol. I. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 501–506.

Carpenter, G.A., Grossberg, S. & Reynolds, J.H. (1991a) ARTMAP: supervised real-time learning and classification by a self-organizing neural network. *Neural Networks*, **4**, 565–588.

Carpenter, G.A., Grossberg, S. & Rosen, D.B. (1991b) Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, **4**, 759–771.

Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H. & Rosen, D.B. (1992) Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, **3**, 698–713.

Craven, M.W. & Shavlik, J.W. (1993) Learning symbolic rules using artificial neural networks. In *Proceedings of the 10th International Machine Learning Conference*, Amherst, MA. San Mateo, CA: Morgan Kaufmann, pp. 73–80.

Craven, M.W. & Shavlik, J.W. (1994) Understanding neural networks via rule extraction and pruning. In M.C. Mozer, P. Smolensky, D.S. Touretzky, J.L. Elman & A.S. Weigend (Eds), *Proceedings of the 1993 Connectionist Models Summer School*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 184–191.

Fu, L.M. (1992) A neural network model for learning rule-based systems. In *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, MD, Vol. I. Piscataway, NJ: IEEE Service Center, pp. 343–348.

Grossberg, S. (1973) Contour enhancement, short term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics*, **52**, 217–257.

Hartigan, J.A. (1975) *Clustering Algorithms*. New York, NY: Wiley.

Kosko, B. (1986) Fuzzy entropy and conditioning. *Information Science*, **40**, 165–174.

Moore, B. (1989) ART 1 and pattern clustering. In *Proceedings of the 1988 Connectionist Models Summer School*, pp. 174–185.

Murphy, P.M. & Aha, D.W. (1992) UCI repository of machine learning databases [machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science.

Nowlan, S.J. & Hinton, G.E. (1992) Simplifying neural networks by soft weight-sharing. *Neural Computation*, **4**, 473–493.

Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Salzberg, S.L. (1990) *Learning with Nested Generalized Exemplars*. Hingham, MA: Kluwer Academic.

Schlimmer, J.S. (1987) Concept acquisition through representational adjustment. PhD thesis,