Rule Extraction: From Neural Architecture to Symbolic Representation

Gail A. Carpenter* and Ah-Hwee Tan†

Center for Adaptive Systems
and
Department of Cognitive and Neural Systems
Boston University
111 Cummington Street
Boston, Massachusetts 02215, USA
Email: gail@cns.bu.edu/atan@cns.bu.edu

Running head: Rule Extraction
Keywords: Fuzzy ARTMAP, rule, confidence factor, pruning.

February 2, 1994

*Supported in part by ARPA (ONR-N00014-92-J-4015), the National Science Foundation (NSF-IRI-90-00530), and the Office of Naval Research (ONR-N00014-91-J-4100).

†Supported by a graduate fellowship from the Institute of Systems Science, National University of Singapore.

Abstract

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, that simplifies the network structure by removing excessive recognition categories and weights; and quantization of continuous learned weights, that allows the final system state to be translated into a usable set of descriptive rules. Rule extraction methods are illustrated in three benchmark studies: (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 backpropagation network, and the *C4.5* decision tree. The ARTMAP rule extraction procedure is also compared with the Knowledgetron and NOFM algorithms, that extract rules from backpropagation networks. Simulation results consistently indicate that ARTMAP rule extraction produces compact sets of comprehensible rules with performance that is comparable, if not superior, to rules extracted by alternative algorithms in terms of both predictive accuracy and system complexity.

Contents

List of Figures

1 Introduction: Rules and Fuzzy ARTMAP

Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds, and Rosen, 1992) is a neural network architecture that performs incremental supervised learning of recognition categories (pattern classes) and multidimensional maps of both binary and analog 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 compatible with if-then rules which link sets of antecedents to their consequences. At any point during the incremental learning process, the system architecture can be translated into a compact set of rules analyzable by human experts. This paper describes such a procedure for translating fuzzy ARTMAP network architecture into a compact rule-based representation.

Rules can be derived more readily from an ARTMAP network than from a backpropagation network, in which the roles of hidden units are usually not explicit. In a fuzzy ARTMAP network, 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 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. In order to describe the knowledge in simplified rule form, real-valued weights are quantized into a small set of values.

The ARTMAP rule extraction algorithm has been evaluated using a Pima Indian diabetes (PID) data set obtained from the UCI repository of machine learning databases (Murphy and Aha, 1992). Simulation results (Carpenter and Tan, 1993) show that pruning consistently produces rule sets that are more accurate than the full system but 1/3 the size. Quantization produces more comprehensible rules at only a slight cost in terms of performance.

In addition to the Pima Indian diabetes benchmark, this paper reports two other benchmark studies that compare ARTMAP rule extraction with two other neural network rule extraction algorithms. Performance is assessed in terms of predictive accuracy and system complexity. Predictive accuracy is measured by the performance of the extracted rules on an unseen test set. System complexity is measured by the number of rules and antecedents in the rule set.

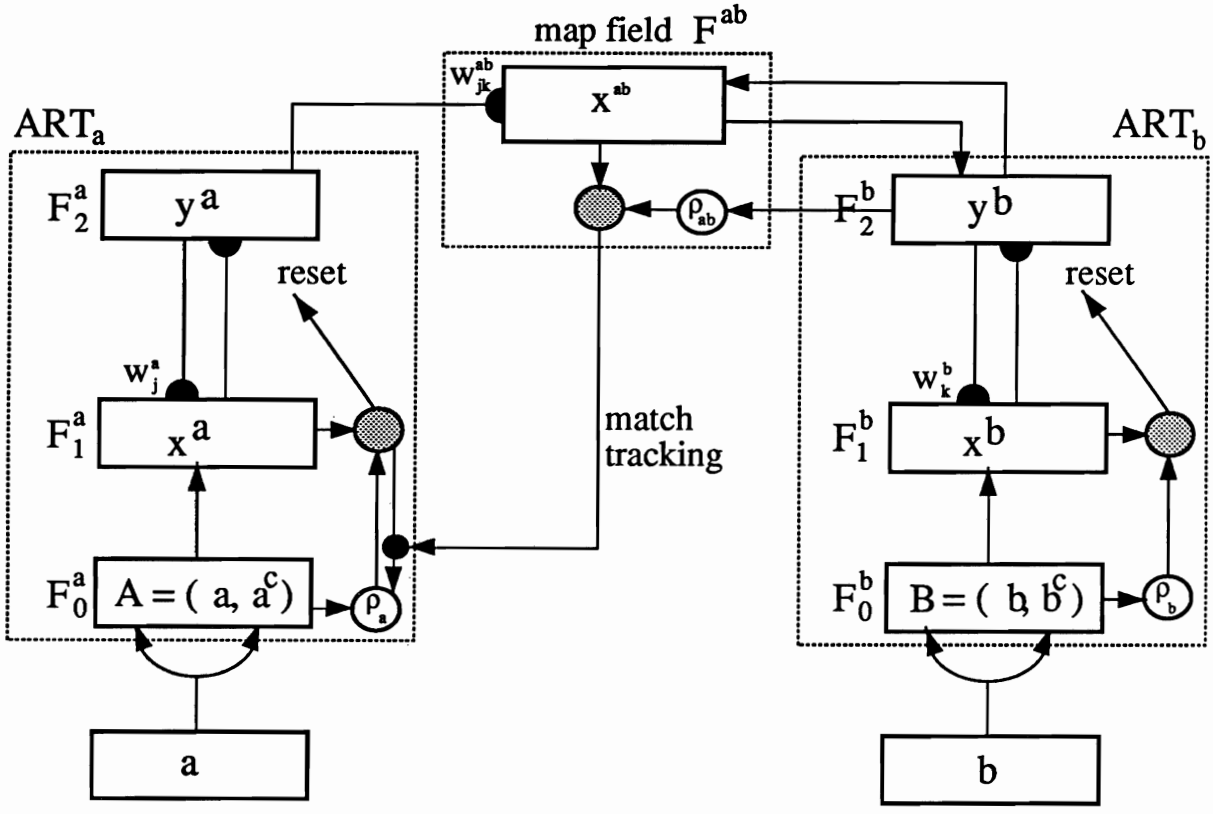

Figure 1: Fuzzy ARTMAP architecture. The ART_a complement coding preprocessor transforms the M_a-vector $\mathbf{a}$ into the $2M_a$-vector $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$ at the ART_a field F_0^a. $\mathbf{A}$ is the input vector to the ART_a field F_1^a. Similarly, the input to F_1^b is the $2M_b$-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.

The first benchmark problem partitions mushroom feature vectors into two classes, edible or poisonous, using a mushroom data set (Schlimmer, 1987). ARTMAP rule extraction is compared with a backpropagation 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 second 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 Neighbor system, the backpropagation network, and the *C4.5* decision tree. The ARTMAP rule extraction procedure is also compared with the NOFM algorithm that extracts rules from backpropagation networks (Craven and 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 the NOFM rules had better code compression. In this paper, a new ARTMAP rule pruning strategy and an antecedent pruning procedure further reduce the complexity of ARTMAP rules. Using the revised pruning method on the 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 ARTMAP. Section 3 describes the ARTMAP rule extraction algorithm. Section 4 reviews the Knowledgetron system and the NOFM algorithm. The final section reports simulation results of the Pima Indian diabetes, mushroom, and DNA promoter benchmark studies.

2 Fuzzy ARTMAP

Fuzzy ARTMAP is a generalization of binary ARTMAP system (Carpenter, Grossberg, and Reynolds, 1991) that learns to classify inputs by a fuzzy set of features, or a pattern of fuzzy membership values between 0 and 1 indicating the extent to which each feature is present. This generalization is accomplished by replacing the ART 1 modules (Carpenter and Grossberg, 1987, 1991) of the binary ARTMAP system with Fuzzy ART modules (Carpenter, Grossberg, and Rosen, 1991). 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 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

ART 1 (BINARY)	Fuzzy ART (ANALOG)
CATEGORY CHOICE	
$T_j = \frac{\lvert \mathbf{I} \cap \mathbf{w}_j \rvert}{\alpha + \lvert \mathbf{w}_j \rvert}$	$T_j = \frac{\lvert \mathbf{I} \wedge \mathbf{w}_j \rvert}{\alpha + \lvert \mathbf{w}_j \rvert}$
MATCH CRITERION	
$\frac{\lvert \mathbf{I} \cap \mathbf{w}_J \rvert}{\lvert \mathbf{I} \rvert} \geq \rho$	$\frac{\lvert \mathbf{I} \wedge \mathbf{w}_J \rvert}{\lvert \mathbf{I} \rvert} \geq \rho$
FAST LEARNING	
$\mathbf{w}_J^{(new)} = \mathbf{I} \cap \mathbf{w}_J^{(old)}$	$\mathbf{w}_J^{(new)} = \mathbf{I} \wedge \mathbf{w}_J^{(old)}$
$\cap$ = logical AND intersection	$\wedge$ = fuzzy AND minimum

Figure 2: Analogy between ART 1 and fuzzy ART.

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 that 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 $\overline{\rho_a}$ 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 $\overline{\rho_a}$ enable larger categories to form, maximizing code compression. Initially, $\rho_a = \overline{\rho_a}$. 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* (Carpenter, Grossberg, and Reynolds, 1991). 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 (Carpenter, Grossberg & Rosen, 1991) incorprates computations from fuzzy set theory into ART systems. The crisp (nonfuzzy) intersection operator ($\cap$) that describes ART 1 dynamics (Carpenter and Grossberg, 1987) 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.

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 $\mathbf{I} = (I_1, \ldots, I_M)$ 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 is arbitrary.

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

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

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.

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 $\wedge$ (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 J denotes the chosen category, where

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

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 J^{th} category is chosen, $y_J = 1$; and $y_j = 0$ for $j \neq J$. In a choice system, the F_1 activity vector $\mathbf{x}$ obeys the equation

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

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:

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho; \tag{7}$$

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

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_J| \geq \rho |\mathbf{I}|. \tag{8}$$

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

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} < \rho; \tag{9}$$

that is, if

$$|\mathbf{x}| = |\mathbf{I} \wedge \mathbf{w}_J| < \rho|\mathbf{I}|. \tag{10}$$

Then the value of the choice function T_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 J represents the active category, selected by (5). The search process continues until the chosen J satisfies the matching criterion (7).

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

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

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.

Fast-commit, slow-recode: For efficient coding of noisy input sets, it is useful to set $\beta = 1$ when 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 $\mathbf{w}_J^{(\text{new})} = \mathbf{I}$ the first time category 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.

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 $\mathbf{a}$ (Figure 1). In its simplest form, $\mathbf{a}$ represent the on-response and $\mathbf{a}^c$, the complement of $\mathbf{a}$, represents the off-response, where

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

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

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

A complement coded input is automatically normalized, because

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

With complement coding, the initial condition

$$w_{j1}(0) = \ldots = w_{j,2M}(0) = 1. \tag{17}$$

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 measures the degree to which $\mathbf{w}_j$ is a fuzzy subset of $\mathbf{I}$ (Kosko, 1986). A category J for which $\mathbf{w}_J$ is a fuzzy subset of $\mathbf{I}$ will then be selected first, if such a category exists. Resonance depends on the degree to which $\mathbf{I}$ is a fuzzy subset of $\mathbf{w}_J$, by (7) and (9). When J is such a fuzzy subset choice, then the match function value is:

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} = \frac{|\mathbf{w}_J|}{|\mathbf{I}|}. \tag{18}$$

Choosing 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 $|\mathbf{w}_J|$, 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.

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 $\mathbf{w}_j^a \equiv (w_{j1}^a, w_{j2}^a, \ldots, w_{j,2M_a}^a)$ denotes the j^{th} 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 k^{th} 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 j^{th} 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.

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:

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

By (19), $\mathbf{x}^{ab} = \mathbf{0}$ if $\mathbf{y}^b$ fails to confirm the map field prediction made by $\mathbf{w}_J^{ab}$. 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

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

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

$$|\mathbf{x}^a| = |\mathbf{A} \wedge \mathbf{w}_J^a| < \rho_a|\mathbf{A}|, \tag{21}$$

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 an F_2^a node J with

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

and

$$|\mathbf{x}^{ab}| = |\mathbf{y}^b \wedge \mathbf{w}_J^{ab}| \geq \rho_{ab}|\mathbf{y}^b|. \tag{23}$$

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.

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

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

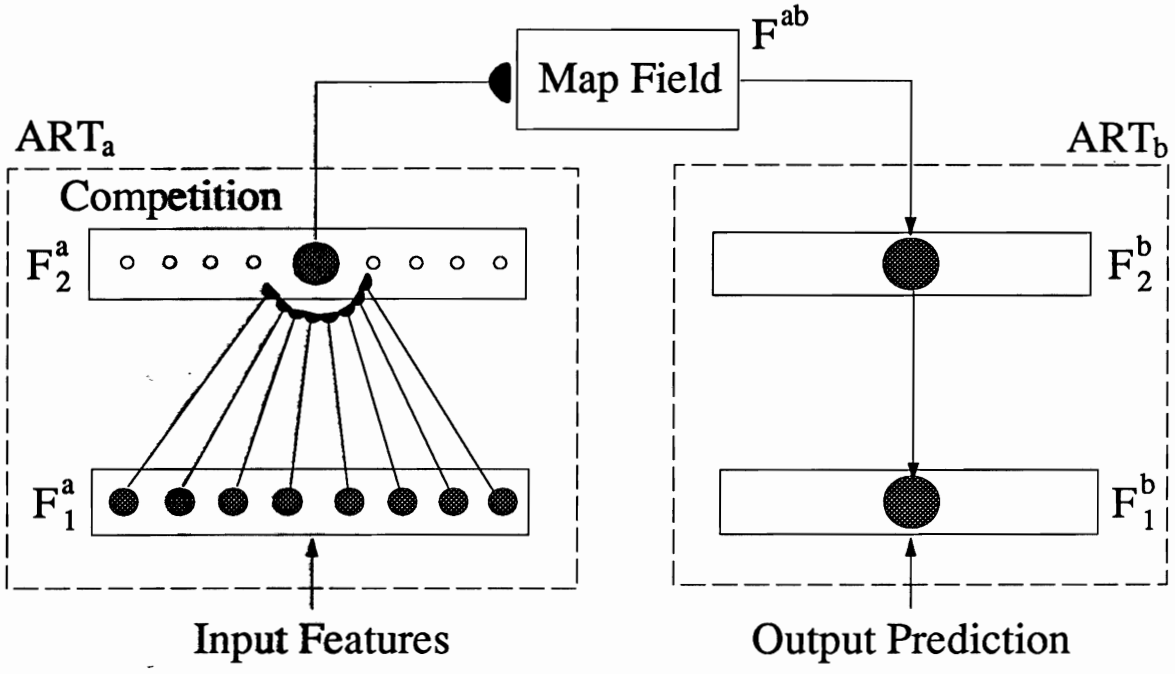

Figure 3: Schematic diagram of a rule in fuzzy ARTMAP. Each F_2^a node maps a prototype feature vector (antecedents) to a prediction (consequence).

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., $w_{JK}^{ab} = 1$ for all time.

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 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 consequences (Figure 3). The number of rules equals the number of F_2^a nodes that become active during learning.

3.1 Pruning

To reduce the complexity of fuzzy ARTMAP, a rule pruning procedure aims to select a small set of rules from trained ARTMAP networks based on their confidence factors. To

derive concise rules, an antecedent pruning procedure aims to remove antecedents from rules while preserving accuracy.

3.1.1 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. The confidence factor identifies good rules with nodes that are frequently and correctly used. This allows pruning of ARTMAP. Pruning removes rules that have low confidence. Overall performance is actually improved when the pruning algorithm removes rules that were created to handle misleading special cases.

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

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

where U_j is the usage of node j, A_j is its accuracy, and $\gamma \in [0, 1]$ is a weighting factor.

For an ART_a category j that predicts outcome k, its usage U_j equals the fraction of training set patterns with outcome k coded by node j (F_j), divided by the maximum fraction of training patterns coded by any node J (F_J):

$$U_j = F_j / max\{F_J\}. \tag{26}$$

For an ART_a category j that predicts outcome k, its accuracy A_j equals the percent of predicting set patterns predicted correctly by node j (P_j), divided by the maximum percent of patterns predicted correctly by any node J (P_J) that predicts outcome k:

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

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 an initial 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.

3.1.2 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.

3.2 Quantizing Weight Values

When learning analog patterns or with 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 $\mathbf{w}_j^a$ are quantized. A *quantization level* Q is defined as the number of feature values used in the extracted fuzzy rules. For example, with $Q = 3$, feature values are described as low, medium, or high in the fuzzy rules. *Quantization by truncation* divides the range of [0,1] into Q intervals and assigns a quantization point to the lower bound of each interval; i.e., for $q = 1 \ldots Q$, let $V_q = (q-1)/Q$. When a weight w falls in interval q, the algorithm reduces the value of w to V_q. *Quantization by round-off* distributes Q quantization points evenly in the range of [0,1], with one at each end point; i.e., for $q = 1 \ldots Q$, let $V_q = (q-1)/(Q-1)$. The algorithm then round-offs a weight w to the nearest V_q value.

4 Backpropagation Rule Extraction

Two algorithms that extract symbolic rules from backpropagation networks are now reviewed. Both algorithms use clustering techniques during training to facilitate rule extraction.

4.1 Knowledgetron

Knowledgetron (Fu, 1992) consists of the Knowledgetron Backpropagation (KTBP) trainer and the Knowledgetron (KT) translator. The KTBP trainer iterates the process of adapting a multi-layer neural network and clustering the 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 turn on the unit. Similarly, the algorithm forms disconfirming rules by exploring combinations of negative attributes and negated positive attributes that turn off the unit.

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 topology and initial weights had been specified by an approximately correct domain theory (Towell and Shavlik, 1992). Its domain was extended by Craven and Shavlik (1993, 1994) who trained backpropagation networks using soft weight-sharing (Nowlan and Hinton, 1992), that encourages weights to form clusters during training. On a 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).

5 Comparative Simulations

5.1 Pima Indian Diabetes Diagnosis (cf: ADAP)

The Pima Indian Diabetes (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 268 (34.9%) are positive, from patients who are diagnosed to be diabetic. The ADAP (adaptive) learning routine (Smith, Everhart, Dickson, Knowler, and Johannes, 1988) has previously been applied to the PID data set. The three-layer ADAP architecture (Figure 4) uses fixed connections from the sensor layer to the association layer, and error feedback to adapt connections from the association layer to the responder layer. Smith *et al.* converted

(1) **Clustering:** The weights converging on each hidden and output unit are grouped into clusters using a standard clustering method, known as the *join* algorithm (Hartigan, 1975).
(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 correctly activate 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.

Table I: A summary of the NOFM algorithm.

the 8 feature values into 37 binary variables. The simulations used 100,000 association units. Real-valued predictions were converted to binary prediction using a selected cutoff. After training on 576 inputs, the *sensitivity* (percent correct of actual positive cases) and *specificity* (percent correct of actual negative cases) on the 192 test set cases were each 76%, using a cutoff of 0.448.

In order to compare fuzzy ARTMAP with ADAP performance, each simulation used the same 576 training set inputs. For extracting rules and evaluating the performance of rules, the 768 cases were partitioned into three subsets, to train, predict, and test. Two partitions, 576/96/96 and 384/192/192, were evaluated. The latter, reported below, yields slightly more stable results. During pruning, the weighting factor γ and the pruning threshold τ were each fixed at 0.5. Two indices are used to compare performance, namely (1) accuracy, equal to the percent correct by binary prediction at 0.5 cutoff; and (2) the c-index. The c-index is a cutoff-independent evaluation of the predictive score, equal to the average probability, over all possible pairs of cases with different outcome, that the classifier will assign a higher score to a positive case. The entire simulation, including the training of fuzzy ARTMAP, extraction of rules, and performance evaluation, was repeated ten times for each method. Using voting strategy, an ARTMAP system was trained in several simulation runs with inputs presented in different orderings. For each test case, voting across 20 simulations produces a net predictive score between 0 to 1.

Voting fuzzy ARTMAP is about as accurate as ADAP but uses far fewer nodes (Table III). With fast learning, ARTMAP learns the 576 training patterns in 6 to 15 input presentations. The reduced rule sets do not classify correctly all the training patterns.

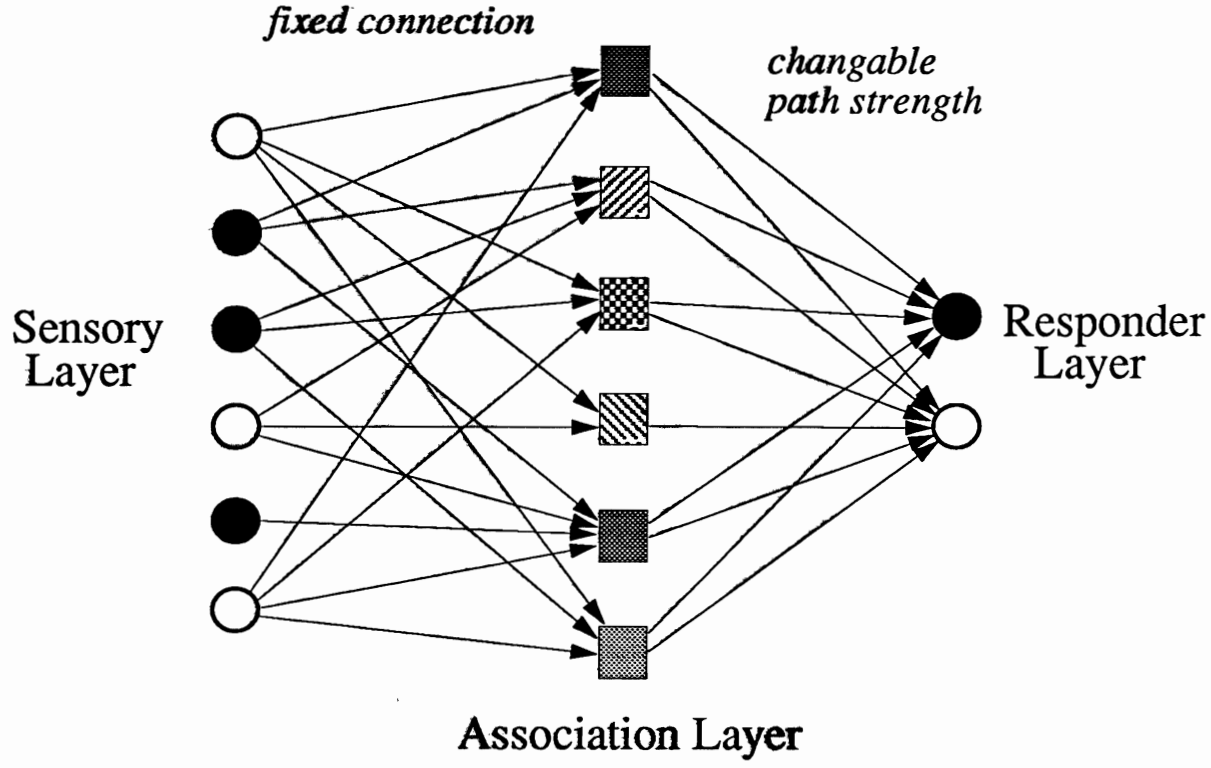

Figure 4: The ADAP architecture.

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 (mu U/ml)	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 II: The eight input features of a PID input and their statistics. The PID data set is obtained from the UCI machine learning repository (Murphy and Aha, 1992).

Methods	Data Set Partition	# Nodes /Rules	Training Acc	Training c-ind	Predicting Acc	Predicting c-ind	Testing Acc	Testing c-ind
ADAP	576/192	100,000	-	-	-	-	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

Table III: 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.

However, reduced overfitting on the training set leads to better performance on the test set (Table III). In particular, the threshold pruning procedure yields about 1/3 as many rules but gives better test set performance in terms of both accuracy and the c-index. Quantization degrades the performance gradually as the number of quantized steps Q decreases. Quantization with $Q = 3$ produces significantly poorer performance across all three subsets. Thus the PID prediction problem cannot be solved by quantized rules using only "low, medium, and high" as feature values. 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 verified and adapted by medical experts. Novel rules discovered through the rule extraction process can be added to expert knowledge and may provide new insights for human and machine diagnosis.

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 nominal features (Table VI). There are 3916 poisonous mushrooms, constituting 48.2% of the total population.

On this problem, Fu (1992) used 1000 inputs to train a backpropagation network con-

Pre-	Feature Weights								Rule Statistics			Testing	
dict	PREG	PGC	DBP	TSFT	SI	BMI	DPF	AGE	Usage	Acc	CF	#	Acc
+	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 IV: Six PID rules extracted by pruning and quantization ($Q = 5$). The pruned set of 23 rules predicts correctly 78.2% train, 77.6% predict, and 76.0% test set vectors. Each rule is described in terms of a set of intervals 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.

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.

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

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	ibrous/scaly/silky/smooth
13	stalk-surface-below-ring	ibrous/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 VI: The 22 input features of the mushroom data set.

Systems	Data Partition	# Rules	# Ante	Train(%)	Predict(%)	Test(%)
Backpropagation	1000/0/1000	-	-	100.0	-	99.0
Knowledgetron	1000/0/1000	233	-	100.0	-	99.6
Fuzzy 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

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

taining 127 input units, 63 hidden units, 2 output units, and 8127 connections. The network classified the 1000 training cases with 100% accuracy and a disjointed 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 nominal 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 the following 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. All simulations are almost 100% 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 cases. Rule pruning and antecedent pruning were applied using the local pruning strategy to reduce the rule set complexity. 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 rules. After pruning, the ARTMAP rule sets, 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.

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

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.

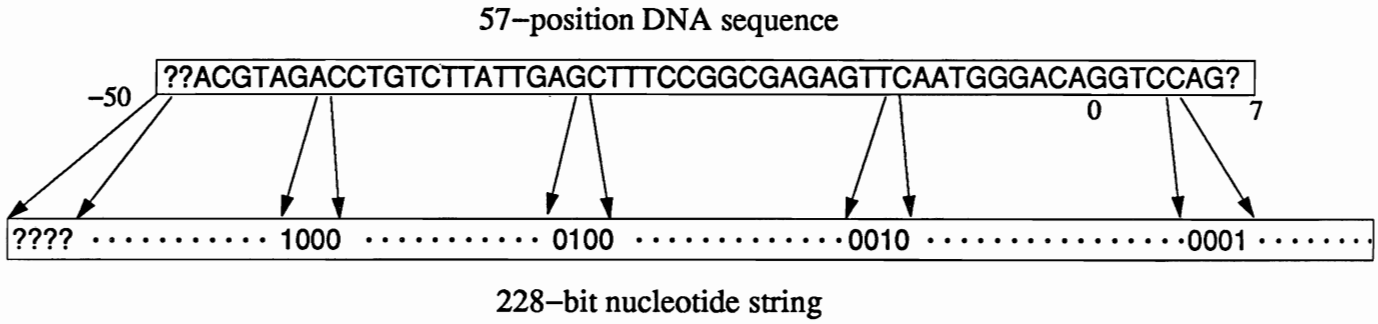

Figure 5: 57-position DNA sequence. Each position takes one of the four nucleotide values (A,G,T,C) or unknown (?). Using local representation, each DNA sequence is expanded into a 228-bit nucleotide string.

5.3 DNA Promoter Recognition (cf: KNN, Backpropagation, 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 and 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. Missing feature values comprise 1% of the total feature population.

In the following sections, fuzzy ARTMAP and ARTMAP rule extraction algorithms are compared with KNN, *C4.5*, backpropagation and NOFM algorithms. All systems handle missing values differently. KNN replaces missing values by ones to obtain a better predictive accuracy. Backpropagation network assigns 0.25 to missing features. ARTMAP ignores features with missing values by replacing them with zeroes.

5.3.1 KNN Simulations

The K Nearest Neighbor (KNN) algorithm is a look-up system that stores all training patterns in its memory. Given a test pattern, its category is determined by a vote of the categories of the K training patterns closest to the test pattern. The test set performance of KNN on the promoter data set was very accurate (Figure 6), with a minimal error

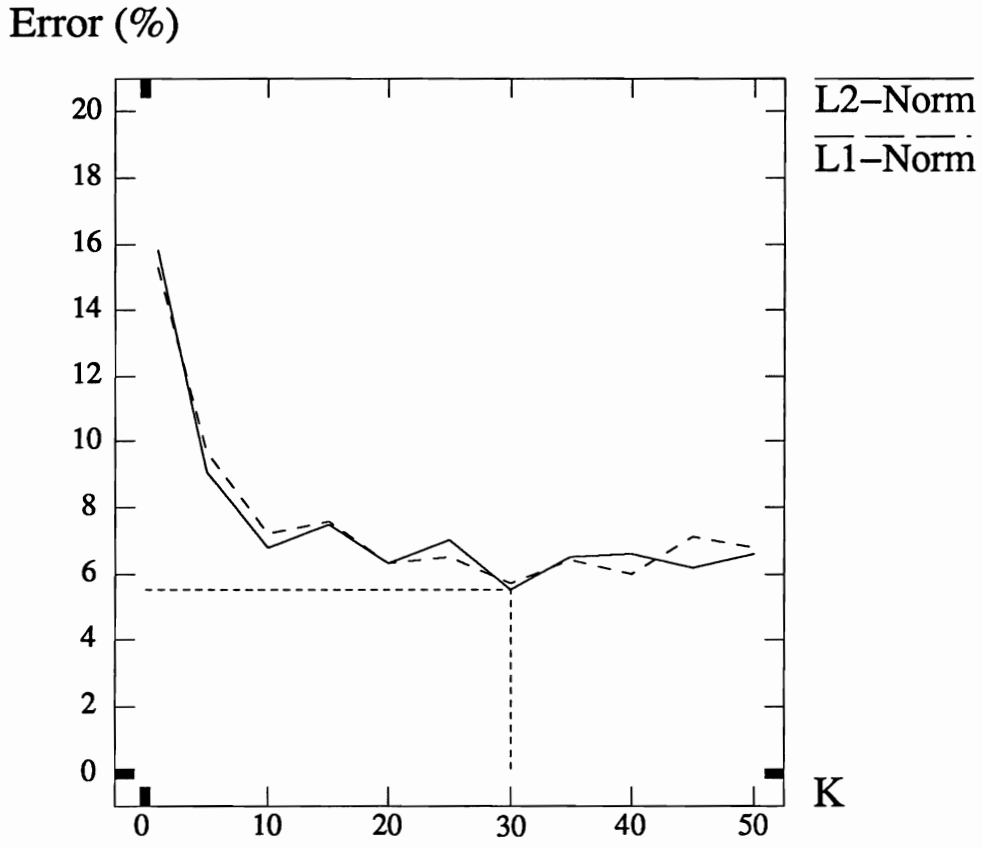

Figure 6: Average predictive error rate of KNN on the promoter data set over 100 runs using $K = 1$ to 50 neighbors.

rate of 5.5% obtained using $K = 30$ neighbors. However, since KNN performs no data compression, it is most useful for problem domains with small data sets. For the promoter data set, KNN stores all 468 patterns with a total of 26,676 attributes. The NOFM and ARTMAP rule extraction simulations reported below create approximately 10 to 20 rules with a total of 100 antecedents, but have error rates around 10%. Therefore, a tradeoff needs to be made between system complexity and predictive accuracy.

5.3.2 Backpropagation, 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 ten-fold cross-validation methodology, the accuracy and the system size of backpropagation network and NOFM rules were compared with those of *C4.5* decision tree and extracted *C4.5* rules.

Backpropagation network trained using soft-weight sharing achieved an error rate of

Systems	# Rules	# Antecedents	Error (%)
C4.5 decision trees	-	-	16.9
C4.5 rules	23.2	47.3	13.5
Backpropagation Networks	-	-	7.9
NOFM rules	8.2	119.6	11.1
NOFM rules (after pruning)	8.1	97.2	10.2

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

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 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 and Ross, 1993), that integrates distributed activity across F_2^a category nodes, is effective in classifying noisy or novel inputs. In ARTMAP systems with category choice, only the F_2^a node J that receives maximal $F_1^a \rightarrow F_2^a$ input T_j^a predicts ART_b output. In simulations,

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

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 $F_1^a \rightarrow F_2^a$ 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 the input T_j^a to the j^{th} F_2^a node to a power p and normalizes the total activity:

$$y_j^a = \frac{(T_j^a)^p}{\sum_k (T_k^a)^p} \tag{29}$$

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

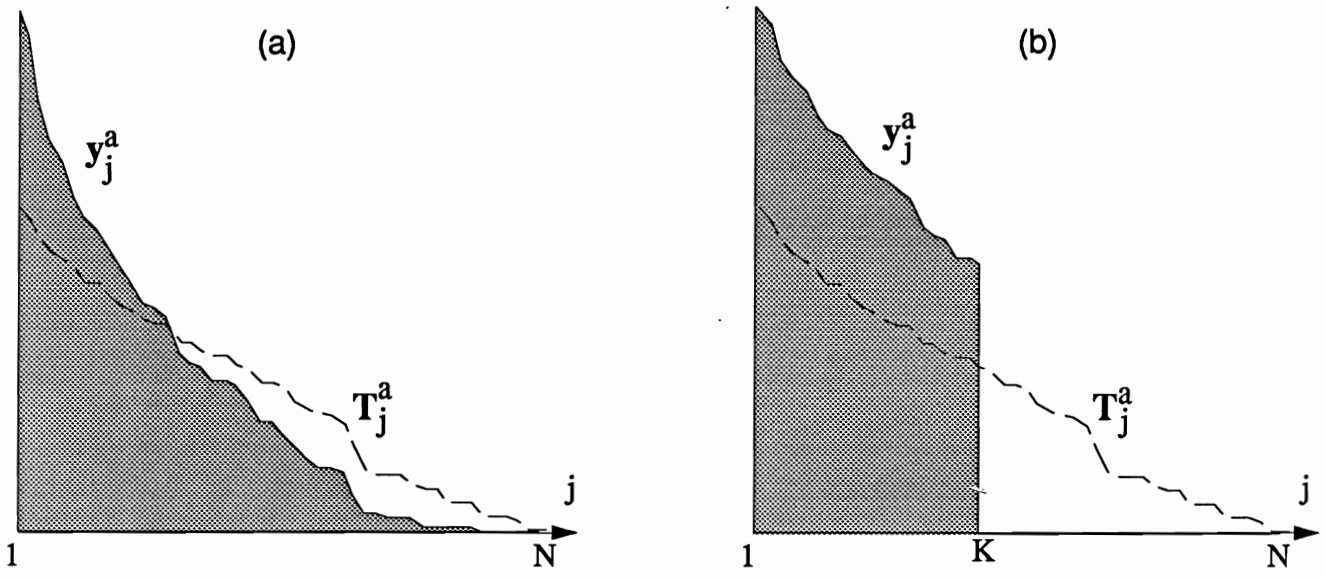

Figure 7: (a) Contrast enhancement by the power rule with $p = 2$. (b) Contrast enhancement by the K-max rule. T_j^a is the input to F_2^a node j. y_j^a is the contrast enhanced activity of node j.

K-max Rule: In the spirit of the K Nearest Neighbor (KNN) system, the K-max rule picks the set of K F_2^a nodes with the largest input T_j^a for prediction. The F_2^a activities y_j^a are then:

$$y_j^a = \begin{cases} \frac{T_j^a}{\sum_{k \in \Phi} T_k^a} & \text{if } j \in \Phi \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

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

After the F_2^a activity vector $\mathbf{y}^a$ is contrast enhanced by (29) or (30), the input $\mathbf{x}^{ab}$ 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 ten-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 left out, using parameter values: $\alpha = 10$, $\beta = 1$, and $\bar{\rho}_a = 0$. The probabilistic score produced

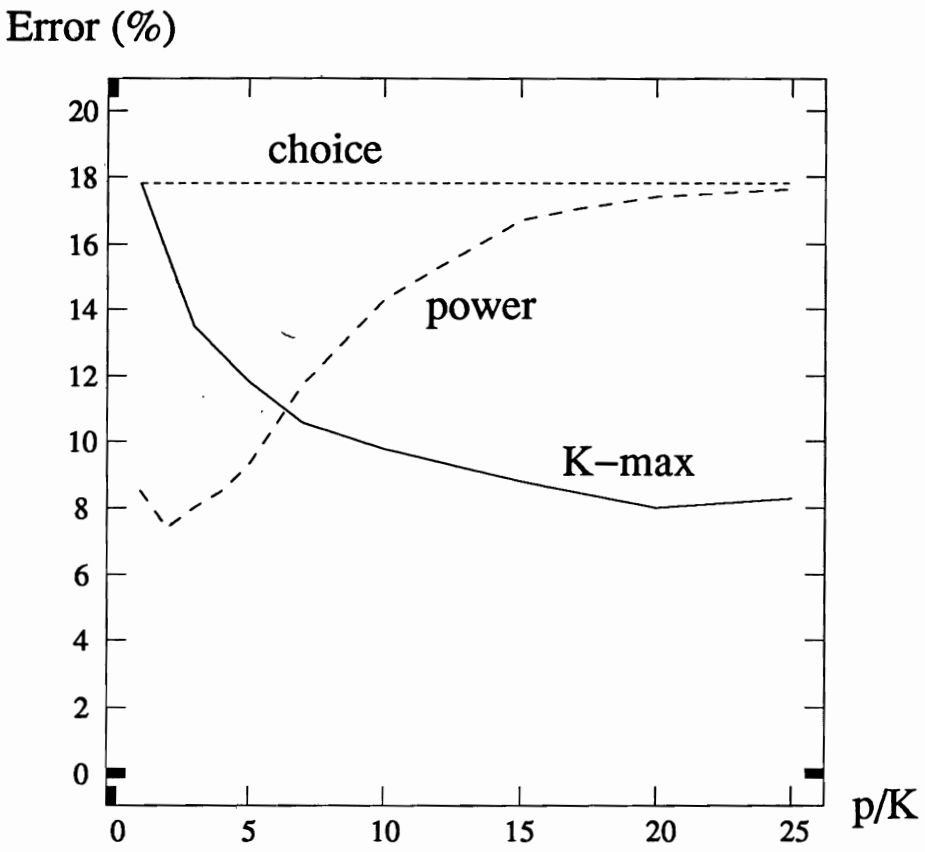

Figure 8: Ten-run average predictive error rate of ART-EMAP on the promoter data set, compared to fuzzy ARTMAP choice at F_2^a. p is the power used in the power rule and K is the number of F_2^a recognition categories used in the K-max rule. With the power rule, compression increases towards choice as $p \to \infty$. With the K-max rule, compression decreases from choice to a linear representation of the input as K goes from 1 to N.

by ART-EMAP spatial evidence accumulation was 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 $K = N$, 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 small p. The best performance of 7.4% error is obtained with $p = 2$ (Figures 7a 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 $p = 2$.

Systems	# Rules	# Antecedents	Error (%)
Backpropagation Networks	-	-	7.9
NOFM rules	8.2	119.6	11.1
NOFM rules (after pruning)	8.1	97.2	10.2
KNN (K=30)	468	26,676	5.5
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

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

5.3.5 ARTMAP Rule Extraction

In ARTMAP rule extraction simulations, seven of the partitions were used to train and evaluate *usage* of each rule; two of the ten input partitions were used to evaluate *accuracy*; and the remaining one to test the extracted rules. Setting the usage/accuracy weighting factor γ (25) equal to 0.4 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 as the input patterns were binary and fast learning was used in simulations.

Table X summarizes ARTMAP performance on the promoter data set together with those obtained by backpropagation, NOFM, and KNN algorithms. The performance of fuzzy ARTMAP is slightly better than that of backpropagation 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 produces a predictive error of 10.4%, which is comparable to that of the NOFM rules. Comparing the system complexity, the NOFM rule sets have fewer rules but have more antecedents than the ARTMAP rule sets.

Also reported are the results obtained with 10-voting ARTMAP. Under the voting strategy, an ARTMAP system is trained in multiple simulation runs using different orderings

of input patterns. The output predictions of ARTMAP across runs are averaged to form a final prediction for each test case. This technique was used by Towell, Shavlik, and Noordewier (1990) in their promoter simulations to obtain a slight improvement in performance. The rule extraction simulations of Craven and Shavlik (1993, 1994) however 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 performs slightly better than the original neural networks. Voting has been a generally useful technique for ARTMAP systems in which fast learning leads to different sets of recognition categories and hence different types of predictive errors across simulations.

5.3.6 Semantic Interpretation and Comparison

Table XI shows a sample set of rules extracted from a fuzzy ARTMAP system. Rules with consequences $P_1, P_2, \ldots, P_6$ are created by positive instances (promoters) and thus are called *positive* rules. Rules with consequences $N_1, N_2, \ldots, N_{10}$ are created by negative instances (non-promoters) and are called *negative* rules. The system prediction is made by the "promoter" rule that emulates ART-EMAP computation, summing evidences of promoter across the positive rules. Note that the positive rules for identifying promoters are quite simple, while the negative rules for identifying non-promoters are slightly more complicated. This is perhaps due to the randomness of 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 and none of them appears in the negative rules. This suggests that these features are good indicators for promoters.

ARTMAP rules are different in form from NOFM rules (Table XII). ARTMAP creates rules for detecting features of both promoters and non-promoters, while NOFM rules focus only on positive instances (promoters), then use a closed-world 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 backpropagation networks. The negative weights in backpropagation networks allow NOFM rules to include negative terms; in ARTMAP rules, the effect of negative terms can be obtained through complement coding of input patterns. Complement coding however is not used in ARTMAP promoter simulations.

Neither system requires an exact match to fire a rule. The NOFM rules are fired based on the satisfaction of individual condition (NOFM) stated within each rule, whereas ARTMAP rule firing is based on competition among all positive and negative rules, on top of the ART_a vigilance (ρ_a) criterion. Considering each rule independently, an ARTMAP

Conseq-uence	Feature Template/ Conditions	Rule Statistics CF	Usage	Acc
Promoter	:- $\sum_{i=1}^{6} f(P_i) > 0.5$ 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 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	:- 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) }.

Table XII: Sample NOFM rules extracted from a backpropagation promoter network (Craven and Shavlik, 1994).

positive or negative rule is roughly equivalent to a $(M \times \rho_a)$-of-M rule. However, when functioning as a whole, each ARTMAP rule affects each other's activities by the ART-EMAP contrast enhancement process. Another feature of ARTMAP rules is that each of them can be assigned a confidence factor which reflects its usefulness and reliability. This information can be very useful to human experts when assigning priorities to rules.

In summary, ARTMAP rules and NOFM rules are roughly comparable in terms of both predictive accuracy and system complexity. Although the two approaches differ in many aspects, both employ certain variants of inexact match construct to maximize code compression. Preserving properties of neural networks in rules thus leads to a more powerful symbolic representation of knowledge.

Acknowledgements

The authors wish to thank Mark Craven and Jude Shavlik for sharing the promoter data set and the NOFM simulation details. We are also 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. and 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. and Grossberg, S. (Ed.) (1991) *Pattern Recognition by Self-Organizing Neural Networks.* Cambridge, MA: MIT Press.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., and 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.

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

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

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

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

Craven, M. W. and Shavlik, J. W. (1994) Understanding neural networks via rule extraction and pruning. *Proceedings, 1993 Connectionist Models Summer School*, pp. 184–191. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

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. *Proceedings, 1988 Connectionist Models Summer School*, pp. 174–185. San Mateo, CA: Morgan Kaufmann.

Murphy, P. M. and 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. and 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. (1990) *Learning with Nested Generalized Examplars.* Hingham, MA: Kluwer Academic.

Salzberg, S. (1991) A nearest hyperrectangle learning method. *Machine Learning*, **6**, 251–276.

Schlimmer, J. S. (1987) *Concept Acquisition Through Representational Adjustment.* PhD thesis, Department of Information and Computer Science, University of California, Irvine.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., and Johannes, R. S. (1988) Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *Proceedings, Symposium on Computer Applications and Medical Care*, pp. 261–265. IEEE Computer Society Press.

Tan, A. H. (1994) Rule learning and extraction with self-organizing neural networks. *Proceedings, 1993 Connectionist Models Summer School*, pp. 192–199. Hillsdale, NJ: Lawrence Erlbaum Associates.

Towell, G. G. and Shavlik, J. W. (1992) Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. *Advances in Neural Information Processing Systems 4*, pp. 977–984. San Mateo, CA: Morgan Kaufmann.

Towell, G. G., Shavlik, J. W., and Noordewier, M. O. (1990) Refinement of approximately correct domain theories by knowledge-based neural networks. *Proceedings, 8th National Conference on AI, Boston, MA*, pp. 861–866. AAAI Press/The MIT Press.

Zadeh, L. (1965) Fuzzy sets. *Information Control*, **8**, 338–353.