ART NEURAL NETS

When developing a neural network to perform a particular pattern-classification task, one typically proceeds by gathering a set of exemplars, or training patterns, and then using these exemplars to train the network. Once the network has adequately learned the exemplars, the weights of the network are fixed, and the system can be used to classify future "unseen" patterns. This operational scenario is acceptable when the problem domain is "well-behaved"—in the sense that it is possible to select a set of training patterns that, once learned, will allow the network to classify future unseen patterns accurately. Unfortunately, in many realistic situations, the problem domain is not well-behaved.

Consider a simple example. Suppose a company wishes to train a neural network to recognize the silhouettes of the parts that are required to produce the products in the company's product line. The appropriate images can be collected and used to train a neural network, a task that is typically computationally time consuming depending on the size of the

J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*. Copyright © 1999 John Wiley & Sons, Inc.

network required. After the network has learned this training set (according to some criteria), the training period is ended and weights are fixed. Now assume that at some future time another product is introduced, and that the company wishes to add the component parts of this new product to the knowledge presently stored in the network. This would typically require a retraining of the network using all of the previous training patterns, plus the new ones. Training on only the new patterns could result in the network learning these new patterns quite well, but forgetting the previously learned patterns. Although this retraining may not take as long as the initial training, it is still likely to require a significant amount of time. Moreover, if the neural network is presented with a previously unseen pattern that is quite different from all of the training patterns, in most neural network models there is no built-in mechanism for recognizing the novelty of the input.

We have been describing what Grossberg calls the *stability–plasticity dilemma* (1). This dilemma can be restated as a series of questions: How can a learning system remain adaptive (plastic) in response to a significant input, yet remain stable in response to an irrelevant input? How does the system know when to switch between the plastic and the stable modes? How can the system retain previously learned information, while continuing to learn new things?

In response to such questions, Grossberg developed the *adaptive resonance theory* (ART) (1). An important element of ART that is used to resolve the stability–plasticity dilemma is the feedback that occurs from the output layer to the input layer of these architectures. This feedback mechanism allows for the learning of new information without destroying old information, the automatic switching between stable and plastic modes, and stabilization of the encoding of the pattern classes. These feedback connections in ART neural network architectures will be clearly illustrated later when these architectures are described in more detail.

Adaptive resonance theory gets its name from the particular way in which learning and recall interplay in these networks. In physics, resonance occurs when a small-amplitude vibration of the proper frequency causes a large-amplitude vibration in an electrical or mechanical system. In an ART network, information in the form of processing-element outputs reverberates back and forth between layers. If the proper patterns develop, a stable oscillation ensues, which is the neural network equivalent of resonance. During this resonant period, learning—or adaptation—can occur. Before the network has achieved a resonant state, no learning takes place, because the time required for changes in the weights is much longer than the time that it takes for the network to achieve resonance.

In ART networks, a resonant state can be attained in one of two ways. If the network has previously learned to recognize an input pattern, then the resonant state will be achieved quickly when the input pattern is presented. During resonance, the adaptation process will reinforce the memory of the stored pattern. If the input pattern is not immediately recognized, the network will rapidly search through its stored patterns looking for a match. If no match is found, the network will enter a resonant state, whereupon a new pattern will be stored for the first time. Thus the network responds quickly to previously learned data, yet remains able to learn when novel data are presented.

Adaptive resonance theory was introduced by Grossberg in 1976 as a means of describing how recognition categories are self-organized in neural networks (1). Since this time, a number of specific neural network architectures based on ART have been proposed. Many of these architectures originated from Carpenter, Grossberg, and their colleagues at Boston University. The first ART neural network architecture, named ART1, appeared in the literature in 1987 (2). This model is an unsupervised neural network capable of self-organizing (clustering) arbitrary collections of binary input patterns. Later in 1987 the ART2 neural network architecture was introduced. This architecture is capable of clustering arbitrary collections of real-valued input patterns (3). The ART2 network was made obsolete in 1991, when the simpler Fuzzy ART architecture was proposed (4). Like ART2, Fuzzy ART is able to cluster real-valued input patterns. In addition, for binary-valued inputs, the operation of Fuzzy ART reduces to that of ART1.

The ART1, ART2, and Fuzzy ART architectures all perform unsupervised learning. In unsupervised learning (also called self-organization), training patterns of unknown classification are used, and there is no external teaching procedure. An internal teaching function determines how network parameters are adapted based on the nature of the input patterns. In this case, the teaching procedure results in the internal categorization of training patterns according to some measure of similarity among the patterns. That is, similar training patterns are grouped together during the training of the network. These groups (or clusters) are then considered to be the pattern classes into which unknown input patterns are later classified.

Supervised learning, on the other hand, requires a set of training patterns of known classification and an external teaching procedure. The teaching procedure is used to adapt network weights according to the network's response to the training patterns. Normally, this adjustment is in proportion to the amount of error present while attempting to classify the current input pattern. The use of supervised learning can logically be separated into two phases—a *training phase* and a *performance phase.* In the training phase, a training set is formed from representative samples taken from the environment in which the neural network is expected to operate. This training set should include sample patterns from all the pattern classes being categorized. Next, the training patterns are applied to the network inputs and the external teacher modifies the system through the use of a training algorithm. Once acceptable results have been obtained from the learning phase, the network may be used in the performance phase. In the performance phase, an unknown pattern is drawn from the environment in which the network operates and applied to the network inputs. At this point, the neural network is expected to perform the recognition task for which it has been trained. If the neural network is able to correctly classify with a high probability input patterns that do not belong to the training set, then it is said to *generalize.* Generalization is one of the most significant concerns when using neural networks to perform pattern classification.

A number of ART architectures have been introduced by the Boston University group of researchers for performing supervised learning. These include ARTMAP (5), in which the input patterns must be binary, and Fuzzy ARTMAP (6), ARTEMAP (7), Gaussian ARTMAP (8), and ARTMAP-IC (9),

where the input patterns can be real valued. The primary purpose of the last three contributions to the supervised-ART family is to improve the generalization performance of Fuzzy ARTMAP.

In conjunction with the vigorous activity of researchers at Boston University in developing ART architectures, other researchers in the field independently developed, analyzed, and applied ART architectures or ART-like architectures to a variety of problems. A short, and obviously not exhaustive, list of such efforts includes the adaptive fuzzy leader clustering (AFLC) (10), LAPART (11), the integrated adaptive fuzzy clustering (IAFC) (12), the Fuzzy Min-Max (13,14), and the Adaptive Hamming Net (15).

In the original ART1 paper (2), a significant portion of the paper is devoted to the analysis of ART1 and its learning properties. Other noteworthy contributions to the analysis and understanding of the learning properties in ART1 can be found in Refs. 16–19. The analysis of Fuzzy ART was initially undertaken in Ref. 4; additional results can be found in Refs. 20 and 21. Properties of learning in the ARTMAP architecture are discussed in Refs. 22 and 23, while properties of learning in the Fuzzy ARTMAP architecture are considered in Ref. 21.

From the discussion above, it is evident that the most fundamental ART architectures are Fuzzy ART and Fuzzy ARTMAP (since the binary versions, ART and ARTMAP, respectively, can be considered special cases). Hence the next four sections of this chapter are devoted to the description of these fundamental ART architectures. We start with Fuzzy ART, because it is the building block for the creation of the Fuzzy ARTMAP architecture. In particular, we discuss in detail the Fuzzy ART architecture, the operation of the Fuzzy ART architecture, and the operating phases (training and performance) of the Fuzzy ART architecture. Next, we discuss the Fuzzy ARTMAP architecture, the operation of the Fuzzy ARTMAP architecture, and the operating phases (training and performance) of the Fuzzy ARTMAP architecture. Later, we present a geometrical interpretation of how Fuzzy ART and Fuzzy ARTMAP operate. This gives a clearer (pictorial) explanation of how these two architectures function. Furthermore, we illustrate with simple examples the training phases of the Fuzzy ART and Fuzzy ARTMAP architectures. A number of applications that make use of ART neural network architectures are considered. Finally, properties of learning in ART1, Fuzzy ART, and ARTMAP are discussed.

FUZZY ART

A brief overview of the Fuzzy ART architecture is provided in the following sections. For a more detailed discussion of this architecture, the reader should consult Ref. 4.

Fuzzy ART Architecture

The Fuzzy ART neural network architecture is shown in Fig. 1. It consists of two subsystems, the *attentional subsystem,* and the *orienting subsystem.* The attentional subsystem consists of two fields of nodes denoted F_1^a and F_2^a. The F_1^a field is called the *input field* because input patterns are applied to it. The F_2^a field is called the *category* or *class representation field* because it is the field where category representations are formed. These categories represent the clusters to which the input patterns belong. The orienting subsystem consists of a

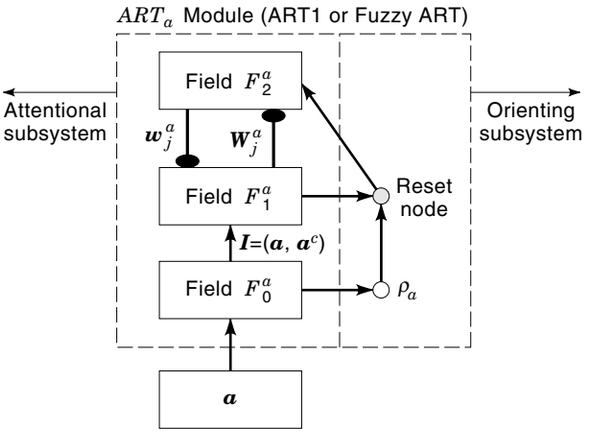

Figure 1. Block diagram of the ART1 or Fuzzy ART architecture.

single node (called the *reset node*), which accepts inputs from the F_1^a field, the F_2^a field (this input is not shown in Fig. 1), and the input pattern applied across the F_1^a field. The output of the reset node affects the nodes in the F_2^a field.

Some preprocessing of the input patterns of the pattern clustering task takes place before they are presented to Fuzzy ART. The first preprocessing stage takes as an input an M_a-dimensional input pattern from the pattern clustering task and transforms it into an output vector $\boldsymbol{a} = (a_1, \ldots, a_{M_a})$, whose every component lies in the interval [0, 1] (i.e., $0 \leq a_i \leq 1$ for $1 \leq i \leq M_a$). The second preprocessing stage accepts as an input the output $\boldsymbol{a}$ of the first preprocessing stage and produces an output vector $\boldsymbol{I}$, such that

$$\boldsymbol{I} = (\boldsymbol{a}, \boldsymbol{a}^c) = \left(a_1, \ldots, a_{M_a}, a_1^c, \ldots, a_{M_a}^c\right) \tag{1}$$

where

$$a_i^c = 1 - a_i \qquad 1 \leq i \leq M_a \tag{2}$$

The above transformation is called *complement coding.* The complement coding operation is performed in Fuzzy ART at a preprocessor field designated by F_0^a (see Fig. 1). We will refer to the vector $\boldsymbol{I}$ formed in this fashion as the *input pattern.*

We denote a node in the F_1^a field by the index i ($i \in \{1, 2, \ldots, 2M_a\}$), and a node in the F_2^a field by the index j ($j \in \{1, 2, \ldots, N_a\}$). Every node i in the F_1^a field is connected via a bottom-up weight to every node j in the F_2^a field; this weight is denoted W_{ij}^a. Also, every node j in the F_2^a field is connected via a top-down weight to every node i in the F_1^a field; this weight is denoted w_{ji}^a. The vector whose components are equal to the top-down weights emanating from node j in the F_2^a field is designated $\boldsymbol{w}_j^a$ and is referred to as a *template.* Note that $\boldsymbol{w}_j^a = (w_{j1}^a, w_{j2}^a, \ldots, w_{j,2M_a}^a)$ for $j = 1, \ldots, N_a$. The vector of bottom-up weights converging to a node j in the F_2^a field is designated $\boldsymbol{W}_j^a$. Note that in Fuzzy ART the bottom-up and top-down weights corresponding to a node j in F_2^a are equal. Hence, in the forthcoming discussion, we will primarily refer to the top-down weights of the Fuzzy ART architecture. Initially, the top-down weights of Fuzzy ART are chosen to be equal to the "all-ones" vector. The initial top-down weight choices in Fuzzy ART are the values of these weights prior to presentation of any input pattern.

Before proceeding, it is important to introduce the notations $\boldsymbol{w}_j^{a,o}$ and $\boldsymbol{w}_j^{a,n}$. Quite often, templates in Fuzzy ART are discussed with respect to an input pattern $\boldsymbol{I}$ presented at the F_1^a field. The notation $\boldsymbol{w}_j^{a,o}$ denotes the template of node j in the F_2^a field of Fuzzy ART *prior* to the presentation of $\boldsymbol{I}$. The notation $\boldsymbol{w}_j^{a,n}$ denotes the template of node j in F_2^a *after* the presentation of $\boldsymbol{I}$. Similarly, any other quantities defined with superscripts $\{a, o\}$ or $\{a, n\}$ will indicate values of these quantities prior to or after a pattern presentation to Fuzzy ART, respectively.

Operation of Fuzzy ART

As mentioned previously, we will use $\boldsymbol{I}$ to indicate an input pattern applied at F_1^a, and $\boldsymbol{w}_j^a$ to indicate the template of node j in F_2^a. In addition, we will use $|\boldsymbol{I}|$ and $|\boldsymbol{w}_j^a|$ to denote the size of $\boldsymbol{I}$ and $\boldsymbol{w}_j^a$, respectively. The size of a vector in Fuzzy ART is defined to be the sum of its components. We define $\boldsymbol{I} \wedge \boldsymbol{w}_j^a$ to be the vector whose ith component is the minimum of the ith $\boldsymbol{I}$ component and the ith $\boldsymbol{w}_j^a$ component. The operation $\wedge$ is called the *fuzzy-min* operation, while a related operation designated by $\vee$ is called the *fuzzy-max* operation. These operations are shown in Fig. 2 for 2 two-dimensional vectors, denoted by $\boldsymbol{x}$ and $\boldsymbol{y}$.

Let us assume that an input pattern $\boldsymbol{I}$ is presented at the F_1^a field of Fuzzy ART. The appearance of pattern $\boldsymbol{I}$ across the F_1^a field produces bottom-up inputs that affect the nodes in the F_2^a field. These bottom-up inputs are given by the equation

$$T_j^a(\boldsymbol{I}) = \frac{|\boldsymbol{I} \wedge \boldsymbol{w}_j^{a,o}|}{(\alpha_a + |\boldsymbol{w}_j^{a,o}|)} \tag{3}$$

where α_a, which takes values in the interval $(0, \infty)$, is called the *choice parameter.* It is worth mentioning that if in the above equation $\boldsymbol{w}_j^{a,o}$ is equal to the "all-ones" vector, then this node is referred to as an *uncommitted node;* otherwise, it is referred to as a *committed node.*

The bottom-up inputs activate a competition process among the F_2^a nodes, which eventually leads to the activation of a single node in F_2^a, namely, the node that receives the maximum bottom-up input from F_1^a. Let us assume that node j_m in F_2^a has been activated through this process. The activation of node j_m in F_2^a indicates that this node is considered as a potential candidate by Fuzzy ART to represent the input pattern $\boldsymbol{I}$. The appropriateness of this node is checked by examining the ratio

$$\frac{|\boldsymbol{I} \wedge \boldsymbol{w}_{j_m}^{a,o}|}{|\boldsymbol{I}|} \tag{4}$$

If this ratio is smaller than the vigilance parameter ρ_a, then node j_m is deemed inappropriate to represent the input pattern $\boldsymbol{I}$, and as a result it is reset (deactivated). The parameter ρ_a is set to a prespecified value in the interval [0, 1]. The deactivation process is carried out by the orienting subsystem and, in particular, by the reset node. If a reset happens, another node in F_2^a (different from node j_m) is chosen to represent the input pattern $\boldsymbol{I}$; the deactivation of a node (nodes) lasts for the entire input pattern presentation. The above process continues until an appropriate node in F_2^a is found, or until all the nodes in F_2^a have been considered. If a node in F_2^a is found appropriate to represent the input pattern $\boldsymbol{I}$, then learning ensues according to the following rules.

Assuming that node j_m has been chosen to represent $\boldsymbol{I}$, the corresponding top-down weight vector $\boldsymbol{w}_{j_m}^{a,o}$ becomes equal to $\boldsymbol{w}_{j_m}^{a,n}$, where

$$\boldsymbol{w}_{j_m}^{a,n} = (\boldsymbol{I} \wedge \boldsymbol{w}_{j_m}^{a,o}) \tag{5}$$

It is worth mentioning that in Eq. (5) we might have $\boldsymbol{w}_{j_m}^{a,n} = \boldsymbol{w}_{j_m}^{a,o}$; in this case we say that no learning occurs for the weights of node j_m. Also note that Eq. (5) is actually a special case of the learning equations of Fuzzy ART that is referred to as *fast learning* (4). In this chapter we only consider the fast learning case. We say that node j_m has *coded* input pattern $\boldsymbol{I}$ if during $\boldsymbol{I}$'s presentation at F_1^a, node j_m in F_2^a is chosen to represent $\boldsymbol{I}$, and the j_m top-down weights are modified as Eq. (5) prescribes. Note that the weights converging to or emanating from an F_2^a node other than j_m (the chosen node) remain unchanged during $\boldsymbol{I}$'s presentation.

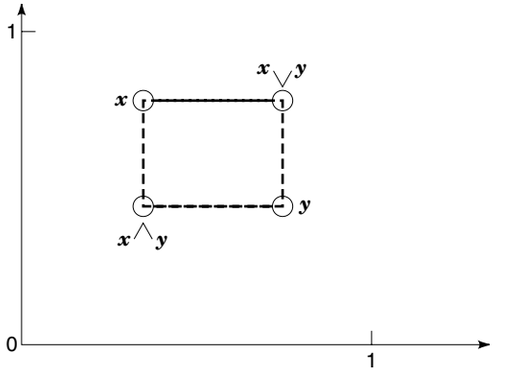

Figure 2. Illustration of the fuzzy min ($\wedge$) and the fuzzy max ($\vee$) operations in the two-dimensional space.

Operating Phases of Fuzzy ART

Fuzzy ART may operate in two different phases: the *training phase* and the *performance phase.* The *training phase* is as follows: Given a collection of input patterns $\boldsymbol{I}^1, \boldsymbol{I}^2, \ldots, \boldsymbol{I}^P$ (i.e., the *training list*), we want Fuzzy ART to cluster these input patterns into different categories. Obviously, we expect patterns that are similar to each other to be clustered in the same category. In order to achieve this goal, one must present the training list repeatedly to the Fuzzy ART architecture. We present $\boldsymbol{I}^1$, then $\boldsymbol{I}^2$, and eventually $\boldsymbol{I}^P$; this corresponds to one *list presentation.* We present the training list as many times as is necessary for Fuzzy ART to cluster the input patterns. The clustering task is considered accomplished (i.e., learning is complete) if the weights in the Fuzzy ART architecture do not change during a list presentation. The aforementioned training scenario is called *off-line training,* and its step-by-step implementation is as follows:

Off-Line Training Phase of Fuzzy ART

1. Choose the Fuzzy ART network parameters (i.e., α_a, M_a, ρ_a) and the initial weights (i.e., $\boldsymbol{w}_j^a$).
2. Choose the pth input pattern from the training list.
3. Calculate the bottom-up inputs at the F_2^a field of the ART_a module due to the presentation of the pth input

pattern. These bottom-up inputs are calculated according to Eq. (3). The bottom-up inputs that are actually required include those for all the committed nodes in F_2^a and the uncommited node of the lowest index.

4. Choose the node in F_2^a that is not disqualified and receives the maximum bottom-up input from F_1^a. Assume that this node is the node with index j_m. Check to see whether this node satisfies the vigilance criterion in ART_a [Eq. (4)].
   a. If node j_m satisfies the vigilance criterion, modify the top-down weights emanating from node j_m according to learning equation (5). If this is the last pattern in the training list go to Step 5. Otherwise, go to Step 2, to present the next in sequence input pattern.
   b. If node j_m does not satisfy the vigilance criterion, disqualify this node and go to the beginning of Step 4.
5. After all patterns have been presented once:
   a. If in the previous list presentation at least one component of top-down weight vectors has changed, go to Step 2 and present the first in sequence input pattern.
   b. If in the previous list presentation no weight changes occurred, the learning process is complete.

In the *performance phase* of Fuzzy ART the learning process is disengaged and patterns from a *test list* are presented in order to evaluate the clustering performance of Fuzzy ART. Specifically, an input pattern from the test list is presented to Fuzzy ART. Through the Fuzzy ART operating rules, discussed previously, a node j_m is chosen in F_2^a that is found appropriate to represent the input pattern. Assuming that some criteria exist for determining how well node j_m represents the cluster to which the input pattern presented to Fuzzy ART belongs, we can apply this process to all the input patterns from the test list to determine how well Fuzzy ART clusters them. Of course, our results are heavily dependent on the criteria used to judge the clustering performance of Fuzzy ART. In the following we propose a procedure to judge this performance.

First, train Fuzzy ART with a list of training patterns until the learning process is complete. The assumption made here is that the list of training patterns is labeled; that is, the label (category) of each input pattern in the list is known. After training, assign a label to each committed node formed in the F_2^a field of Fuzzy ART. A committed node formed in F_2^a is labeled by the output pattern to which most of the input patterns that are represented by this node are mapped. The clustering performance of Fuzzy ART is evaluated by presenting to it, one more time, the input patterns from the training list. For each input pattern from the training list, Fuzzy ART chooses a node in F_2^a. If the label of this node is the output pattern to which this pattern corresponds, then we say that Fuzzy ART clustered this input pattern correctly. If, on the other hand, the label of this node is different from the output pattern to which this input pattern corresponds, then we say that Fuzzy ART made an erroneous clustering. The aforementioned procedure for evaluating clustering performance is suggested in Ref. 24.

ART1

The ART1 architecture, operation, and operating phases are identical to those of Fuzzy ART. The only difference being that, in ART1, the input patterns are not complement coded. Hence, in ART1, the preprocessing field F_0^a of Fig. 1 is not needed.

FUZZY ARTMAP

A brief overview of the Fuzzy ARTMAP architecture is provided in the following sections. For a more detailed discussion of this architecture, the reader should consult Ref. 6.

Fuzzy ARTMAP Architecture

A block diagram of the Fuzzy ARTMAP architecture is provided in Fig. 3. Note that two of the three modules in Fuzzy ARTMAP are Fuzzy ART architectures. These modules are designated ART_a and ART_b in Fig. 3. The ART_a module accepts as inputs the input patterns, while the ART_b module accepts as inputs the output patterns of the pattern classification task. All the previous details are valid for the ART_a module without change. These details are also valid for the ART_b module, where the superscript a is replaced with the superscript b. One of the differences between the ART_a and the ART_b modules in Fuzzy ARTMAP is that for pattern classification tasks (many-to-one maps) it is not necessary to apply complement coding to the output patterns presented to the ART_b module.

As illustrated in Fig. 3, Fuzzy ARTMAP contains a module that is designated the inter-ART module. The purpose of this module is to make sure the appropriate mapping is established between the input patterns presented to ART_a, and the output patterns presented to ART_b. There are connections (weights) between every node in the F_2^a field of ART_a, and all nodes in the F_{ab} field of the inter-ART module. The weight vector with components emanating from node j in F_2^a and converging to the nodes of F_{ab} is denoted $\boldsymbol{w}_j^{ab} = (w_{j1}^{ab}, \ldots, w_{jk}^{ab}, \ldots, w_{jN_b}^{ab})$, where N_b is the number of nodes in F_{ab} (the number of nodes in F_{ab} is equal to the number of nodes in F_2^b). There are also fixed bidirectional connections between a node k in F_{ab} and its corresponding node k in F_2^b.

Operation of Fuzzy ARTMAP

The operation of the Fuzzy ART modules in Fuzzy ARTMAP is slightly different from the operation of Fuzzy ART described previously. For one thing, resets in the ART_a module of Fuzzy ARTMAP can have one of two causes: (1) the category chosen in F_2^a does not match the input pattern presented at F_1^a, or (2) the appropriate map has not been established between an input pattern presented at ART_a and its corresponding output pattern presented at ART_b. This latter type of reset, which Fuzzy ART does not have, is enforced by the inter-ART module via its connections with the orienting subsystem in ART_a (see Fig. 3). This reset is accomplished by forcing the ART_a architecture to increase its vigilance parameter value above the level that is necessary to cause a reset of the activated node in the F_2^a field. Hence, in the ART_a module of Fuzzy ARTMAP, we identify two vigilance parameter values, a baseline vigilance parameter value $\bar{\rho}_a$, which is the vigilance parameter of ART_a prior to the presentation of an input/output pair to Fuzzy ARTMAP, and a vigilance parameter ρ_a, which corresponds to the vigilance parameter that is established in ART_a via appropriate resets enforced by the

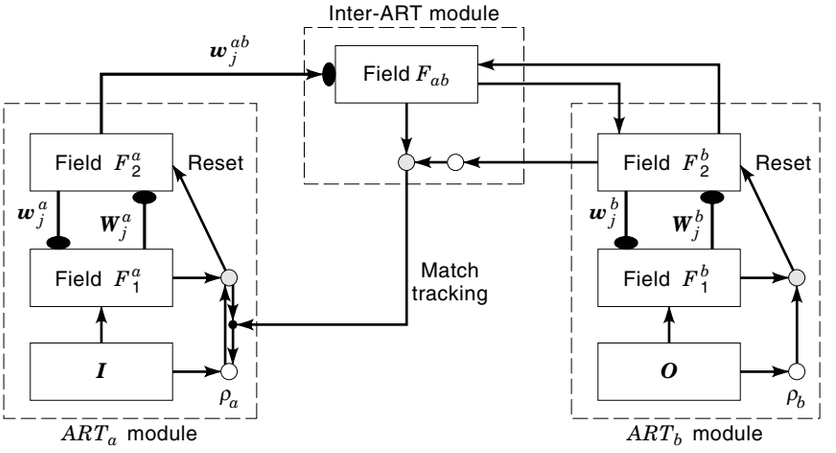

Figure 3. Block diagram of the ARTMAP or Fuzzy ARTMAP architecture.

inter-ART module. Also, the node activated in F_2^a due to a presentation of an output pattern at F_1^b can either be the node receiving the maximum bottom-up input from F_1^b or the node designated by the F_{ab} field in the inter-ART module. The latter type of activation is enforced by the connections between the F_{ab} field and the F_2^b field.

Equations (1)–(5) for the Fuzzy ART module are valid for the ART_a and ART_b modules in Fuzzy ARTMAP. In particular, the bottom-up inputs to the F_2^a field and the F_2^b field are given by

$$T_j^a(\boldsymbol{I}) = \frac{|\boldsymbol{I} \wedge \boldsymbol{w}_j^{a,o}|}{\alpha_a + |\boldsymbol{w}_j^{a,o}|} \tag{6}$$

and

$$T_k^b(\boldsymbol{O}) = \frac{|\boldsymbol{O} \wedge \boldsymbol{w}_k^{b,o}|}{\alpha_b + |\boldsymbol{w}_k^{b,o}|} \tag{7}$$

where in Eq. (7), $\boldsymbol{O}$ stands for the output pattern associated with the input pattern $\boldsymbol{I}$, while the rest of the ART_b quantities are defined as they were defined for the ART_a module. Similarly, the vigilance ratios for ART_a and ART_b are computed as follows:

$$\frac{|\boldsymbol{I} \wedge \boldsymbol{w}_J^{a,o}|}{|\boldsymbol{I}|} \tag{8}$$

and

$$\frac{|\boldsymbol{O} \wedge \boldsymbol{w}_K^{b,o}|}{|\boldsymbol{O}|} \tag{9}$$

The equations that describe the modifications of the weight vectors $\boldsymbol{w}_j^{ab}$ can be explained as follows. A weight vector emanating from a node in F_2^a to all the nodes in F_{ab} is initially the "all-ones" vector and, after training that involves this F_2^a node, all of its connections to F_{ab}, except one, are reduced to the value of zero.

Operating Phases of Fuzzy ARTMAP

The operating phases of Fuzzy ARTMAP are the same as the operating phases of Fuzzy ART, the only difference being that in the training phases of Fuzzy ARTMAP, input patterns are presented along with corresponding output patterns. As is the case with Fuzzy ART, Fuzzy ARTMAP may operate in two different phases: training and performance. Here we focus on classification tasks, where many inputs are mapped to a single, distinct output. It turns out that for classification tasks, the operations performed at the ART_b and inter-ART modules can be ignored, and the algorithm can be described by simply referring to the top-down weights of the ART_a module.

The *training phase* of Fuzzy ARTMAP works is as follows. Given the training list $\{\boldsymbol{I}^1, \boldsymbol{O}^1\}$, $\{\boldsymbol{I}^2, \boldsymbol{O}^2\}$, . . ., $\{\boldsymbol{I}^P, \boldsymbol{O}^P\}$, we want Fuzzy ARTMAP to map every input pattern of the training list to its corresponding output pattern. In order to achieve the aforementioned goal, present the training list repeatedly to the Fuzzy ARTMAP architecture. That is, present $\boldsymbol{I}^1$ to ART_a and $\boldsymbol{O}^1$ to ART_b, then $\boldsymbol{I}^2$ to ART_a and $\boldsymbol{O}^2$ to ART_b, and eventually $\boldsymbol{I}^P$ to ART_a and $\boldsymbol{O}^P$ to ART_b; this corresponds to one *list presentation.* Present the training list as many times as is necessary for Fuzzy ARTMAP to classify the input patterns. The classification (mapping) task is considered accomplished (i.e., the learning is complete) when the weights do not change during a list presentation. The aforementioned training scenario is called *off-line training,* and its step-by-step implementation is as follows:

Off-Line Training Phase of Fuzzy ARTMAP

1. Choose the Fuzzy ARTMAP network parameters (i.e., $M_a, \alpha_a, \bar{\rho}_a$) and the initial weights (i.e., $\boldsymbol{w}_j^a$).
2. Choose the pth input/output pair from the training list. Set the vigilance parameter ρ_a equal to the baseline vigilance parameter $\bar{\rho}_a$.
3. Calculate the bottom-up inputs at the F_2^a field of the ART_a module due to the presentation of the pth input pattern. These bottom-up inputs are calculated according to Eq. (6). When calculating bottom-up inputs at F_2^a, consider all committed nodes in F_2^a and the uncommited node with the lowest index.
4. Choose the node in F_2^a that is not disqualified and receives the maximum bottom-up input from F_1^a. Assume that this node has index j_m. Check to see whether this node satisfies the vigilance criterion in ART_a [see Eq. (8)].

a. If node j_m satisfies the vigilance criterion, go to Step 5.
b. If node j_m does not satisfy the vigilance criterion, disqualify this node, and go to the beginning of Step 4.

5. Now consider three cases:
   a. If node j_m is an uncommitted node, designate the mapping of node j_m to be the output pattern $\boldsymbol{O}^p$. Note that $\boldsymbol{O}^p$ is the output pattern corresponding to the input pattern $\boldsymbol{I}^p$ presented in F_1^a. Also, the top-down weights corresponding to node j_m are modified according to Eq. (5). If this is the last input/output pair in the training list go to Step 6. Otherwise, go to Step 2, to present the next in sequence input/output pair.
   b. If node j_m is a committed node, and due to prior learning node j_m is mapped to an output pattern equal to $\boldsymbol{O}^p$, then the correct mapping is achieved, and the top-down weights corresponding to node j_m are modified according to Eq. (5). If this is the last input/output pair in the training list go to Step 6. Otherwise, go to Step 2, to present the next in sequence input/output pair.
   c. If node j_m is a committed node, and due to prior learning node j_m is mapped to an output pattern different from $\boldsymbol{O}^p$, then the mapping is incorrect, and we disqualify the activated node j_m by increasing the vigilance parameter in ART_a to a level that is sufficient to disqualify node j_m. In particular, the vigilance parameter in ART_a (ρ_a) becomes

$$\frac{\left|\boldsymbol{I} \wedge \boldsymbol{w}_{j_m}^{a,o}\right|}{|\boldsymbol{I}|} + \epsilon \tag{10}$$

   where ϵ is a very small positive quantity. Go to Step 4.

6. After all patterns have been presented once, consider two cases:
   a. In the previous list presentation, at least one component of top-down weight vectors has changed. In this case, go to Step 2, and present the first in sequence input/output pair.
   b. In the previous list presentation, no weight changes occurred. In this case, the learning process is complete.

In the *performance phase* of Fuzzy ARTMAP the learning process is disengaged, and input/output patterns from a *test list* are presented in order to evaluate its classification performance. In particular, during the performance evaluation of Fuzzy ARTMAP, only the input patterns of the test list are presented to the ART_a module. Every input pattern from the test list will choose a node in the F_2^a field. If the output pattern to which the activated node in F_2^a is mapped matches the output pattern to which the presented pattern should be mapped, then Fuzzy ARTMAP classified the test input pattern correctly; otherwise Fuzzy ARTMAP committed a classification error.

ARTMAP

The ARTMAP architecture, operation, and operating phases are identical to those of Fuzzy ARTMAP. The only difference is that the input and output patterns in ARTMAP must be binary vectors.

TEMPLATES IN FUZZY ART AND FUZZY ARTMAP: A GEOMETRICAL INTERPRETATION

We previously referred to the top-down weights emanating from a node in the F_2^a field as a *template.* A template corresponding to a committed node is called a *committed template,* while a template corresponding to an uncommitted node is called *uncommitted template.* As we have already mentioned, an uncommitted template has all of its components equal to one.

In the original Fuzzy ART paper (4), it is demonstrated that a committed template $\boldsymbol{w}_j^a$, which has coded input patterns $\boldsymbol{I}^1 = (\boldsymbol{a}(1), \boldsymbol{a}^c(1))$, $\boldsymbol{I}^2 = (\boldsymbol{a}(2), \boldsymbol{a}^c(2))$, . . ., $\boldsymbol{I}^P = (\boldsymbol{a}(P), \boldsymbol{a}^c(P))$, can be written as

$$\begin{aligned} \boldsymbol{w}_j^a &= \boldsymbol{I}^1 \wedge \boldsymbol{I}^2 \wedge \cdots \wedge \boldsymbol{I}^P = \left(\wedge_{i=1}^P \boldsymbol{a}(i), \wedge_{i=1}^P \boldsymbol{a}^c(i)\right) \\ &= \left(\wedge_{i=1}^P \boldsymbol{a}(i), \{\vee_{i=1}^P \boldsymbol{a}(i)\}^c\right) \end{aligned} \tag{11}$$

or

$$\boldsymbol{w}_j^a = (\boldsymbol{u}_j^a, \{\boldsymbol{v}_j^a\}^c) \tag{12}$$

where

$$\boldsymbol{u}_j^a = \wedge_{i=1}^P \boldsymbol{a}(i) \tag{13}$$

and

$$\boldsymbol{v}_j^a = \vee_{i=1}^P \boldsymbol{a}(i) \tag{14}$$

Based on the aforementioned expression for $\boldsymbol{w}_j^a$, we can now state that the weight vector $\boldsymbol{w}_j^a$ can be expressed in terms of the two M_a-dimensional vectors $\boldsymbol{u}_j^a$ and $\boldsymbol{v}_j^a$. Hence the weight vector $\boldsymbol{w}_j^a$ can be represented, geometrically, in terms of two points in the M_a-dimensional space, $\boldsymbol{u}_j^a$ and $\boldsymbol{v}_j^a$. Another way of looking at it is that $\boldsymbol{w}_j^a$ can be represented, geometrically, in terms of a hyperrectangle R_j^a with endpoints $\boldsymbol{u}_j^a$ and $\boldsymbol{v}_j^a$ (see Fig. 4 for an illustration of this when $M_a = 2$). For simplicity, we refer to *hyperrectangles* as *rectangles* because most of our illustrations are in the two-dimensional space.

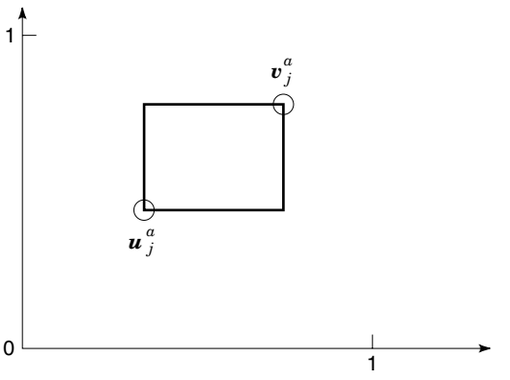

Figure 4. Representation of the template $\boldsymbol{w}_j^a = (\boldsymbol{u}_j^a, \{\boldsymbol{v}_j^a\}^c)$ in terms of the rectangle R_j^a with endpoints $\boldsymbol{u}_j^a$ and $\boldsymbol{v}_j^a$ (in the figure $M_a = 2$).

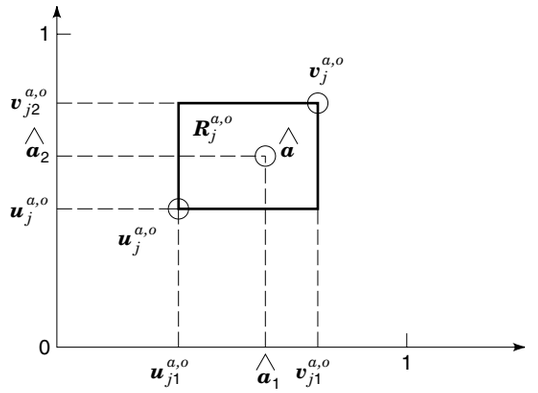

Figure 5. Input pattern $\hat{\boldsymbol{I}} = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}^c)$ represented by the point $\hat{\boldsymbol{a}}$, lies inside rectangle $R_j^{a,o}$ that represents template $\boldsymbol{w}_j^{a,o} = (\boldsymbol{u}_j^{a,o}, \{\boldsymbol{v}_j^{a,o}\}^c)$. Learning of $\hat{\boldsymbol{I}}$ leaves $R_j^{a,o}$ intact.

Obviously, the aforementioned representation implies that we can geometrically represent an input pattern $\boldsymbol{I} = (\boldsymbol{a}, \boldsymbol{a}^c)$ by a rectangle with endpoints $\boldsymbol{a}$ and $\boldsymbol{a}$. In other words, $\boldsymbol{I}$ can be represented by a rectangle of size 0, which is the single point $\boldsymbol{a}$ in the M_a-dimensional space. Note that the *size of a rectangle* R_j^a with endpoints $\boldsymbol{u}_j^a$ and $\boldsymbol{v}_j^a$ is taken to be equal to the norm of the vector $\boldsymbol{v}_j^a - \boldsymbol{u}_j^a$. The *norm of a vector* in Fuzzy ART or Fuzzy ARTMAP is defined to be equal to the sum of the absolute values of its components.

In summary, we will treat $\boldsymbol{w}_j^a = (\boldsymbol{u}_j^a, \{\boldsymbol{v}_j^a\}^c)$ as a rectangle R_j^a with endpoints $\boldsymbol{u}_j^a$ and $\boldsymbol{v}_j^a$ in the M_a-dimensional space, and $\boldsymbol{I} = (\boldsymbol{a}, \boldsymbol{a}^c)$ as the point $\boldsymbol{a}$ in the M_a-dimensional space.

The reason why the rectangle representation of a template $\boldsymbol{w}_j^a$ is so useful is explained below. Consider the template $\boldsymbol{w}_j^{a,o}$, and its geometrical representative, the rectangle $R_j^{a,o}$ with endpoints $\boldsymbol{u}_j^{a,o}$ and $\boldsymbol{v}_j^{a,o}$. Assume that $\boldsymbol{u}_j^{a,o} = \wedge_{i=1}^{P} \boldsymbol{a}(i)$ and $\boldsymbol{v}_j^{a,o} = \vee_{i=1}^{P} \boldsymbol{a}(i)$. Let us now present pattern $\hat{\boldsymbol{I}} = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}^c)$ to Fuzzy ART. Recall that the quantities defined above with a superscript $\{a, o\}$ indicate values of these quantities prior to the presentation of $\hat{\boldsymbol{I}}$ to Fuzzy ART. Suppose that, during $\hat{\boldsymbol{I}}$'s presentation to Fuzzy ART, node j in the F_2^a field is chosen and node j with corresponding weight vector $\boldsymbol{w}_j^{a,o}$ is appropriate to represent the input pattern $\hat{\boldsymbol{I}}$. We now distinguish two cases.

In case 1 we assume that $\hat{\boldsymbol{I}}$ lies inside the rectangle $R_j^{a,o}$ that geometrically represents the template $\boldsymbol{w}_j^{a,o}$ (see Fig. 5). According to the Fuzzy ART rules $\boldsymbol{w}_j^{a,o}$ now becomes equal to $\boldsymbol{w}_j^{a,n}$, where

$$\boldsymbol{w}_j^{a,n} = \boldsymbol{w}_j^{a,o} \wedge \hat{\boldsymbol{I}} = (\boldsymbol{u}_j^{a,o} \wedge \hat{\boldsymbol{a}}, \{\boldsymbol{v}_j^{a,o} \vee \hat{\boldsymbol{a}}\}^c) = (\boldsymbol{u}_j^{a,o}, \{\boldsymbol{v}_j^{a,o}\}^c) = \boldsymbol{w}_j^{a,o}$$

In this case there is no actual weight change or, equivalently, the size of the rectangle that represents the template of node j remains unchanged.

In case 2, we assume that $\hat{\boldsymbol{I}}$ lies outside the rectangle $R_j^{a,o}$ that geometrically represents template $\boldsymbol{w}_j^{a,o}$ (see Fig. 6). Once more, according to the Fuzzy ART rules, $\boldsymbol{w}_j^{a,o}$ becomes equal to $\boldsymbol{w}_j^{a,n}$, where

$$\begin{aligned} \boldsymbol{w}_j^{a,n} &= \boldsymbol{w}_j^{a,o} \wedge \hat{\boldsymbol{I}} = \left(\boldsymbol{u}_j^{a,o} \wedge \hat{\boldsymbol{a}}, \{\boldsymbol{v}_j^{a,o} \vee \hat{\boldsymbol{a}}\}^c\right) \\ &= \left(u_{j1}^{a,o} \wedge \hat{a}_1, \ldots, u_{jM_a}^{a,o} \wedge \hat{a}_{M_a}, \left(v_{j1}^{a,o} \vee \hat{a}_1\right)^c, \ldots, \left(v_{jM_a}^{a,o} \vee \hat{a}_{M_a}\right)^c\right) \\ &\neq \left(\boldsymbol{u}_j^{a,o}, \{\boldsymbol{v}_j^{a,o}\}^c\right) = \boldsymbol{w}_j^{a,o} \end{aligned} \tag{15}$$

In this case there is actual weight change; the size of the rectangle that represents the template of node j is now increased. Thus, during the training process of Fuzzy ART or Fuzzy ARTMAP, the size of a rectangle R_j^a, which the weight vector $\boldsymbol{w}_j^a$ defines, can only increase from the size of zero to possibly a maximum size, which will be determined next.

The maximum size of a rectangle is determined by the vigilance parameter ρ_a. More specifically, with complement coding the size of an input pattern $\boldsymbol{I}$ is equal to M_a. Hence a node j in the F_2^a field with corresponding weight vector $\boldsymbol{w}_j^{a,o}$ codes an input pattern $\boldsymbol{I}$ if the following criterion is satisfied:

$$|\boldsymbol{I} \wedge \boldsymbol{w}_j^{a,o}| \geq M_a \rho_a \tag{16}$$

However,

$$\begin{aligned} |\boldsymbol{I} \wedge \boldsymbol{w}_j^{a,o}| &= \left|(\boldsymbol{a}, \boldsymbol{a}^c) \wedge \left(\boldsymbol{u}_j^{a,o}, \{\boldsymbol{v}_j^{a,o}\}^c\right)\right| \\ &= \left|\left(\boldsymbol{a} \wedge \boldsymbol{u}_j^{a,o}, \boldsymbol{a}^c \wedge \{\boldsymbol{v}_j^{a,o}\}^c\right)\right| \\ &= \left|\left(\boldsymbol{a} \wedge \boldsymbol{u}_j^{a,o}, \boldsymbol{a}^c \vee \{\boldsymbol{v}_j^{a,o}\}^c\right)\right| \\ &= \sum_{i=1}^{M_a} \left(a_i \wedge u_{ji}^{a,o}\right) + \sum_{i=1}^{M_a} \left(a_i \vee v_{ji}^{a,o}\right)^c \\ &= \sum_{i=1}^{M_a} \left(a_i \wedge u_{ji}^{a,o}\right) + M_a - \sum_{i=1}^{M_a} \left(a_i \vee v_{ji}^{a,o}\right) \\ &= M_a - \left|\left(\boldsymbol{a} \vee \boldsymbol{v}_j^{a,o}\right) - \left(\boldsymbol{a} \wedge \boldsymbol{u}_j^{a,o}\right)\right| \\ &= M_a - \left|R_j^{a,n}\right| \end{aligned} \tag{17}$$

From the above equations we can see that the rectangle size is allowed to increase provided that the new rectangle size satisfies the constraint

$$\left|R_j^{a,n}\right| \leq M_a(1 - \rho_a) \tag{18}$$

The above inequality implies that if we choose ρ_a small (i.e., $\rho_a \approx 0$), then some of the rectangles that the Fuzzy ART architecture defines might fill most of the entire input pattern space. On the other hand, if ρ_a is close to 1, all of the rectangles will be small.

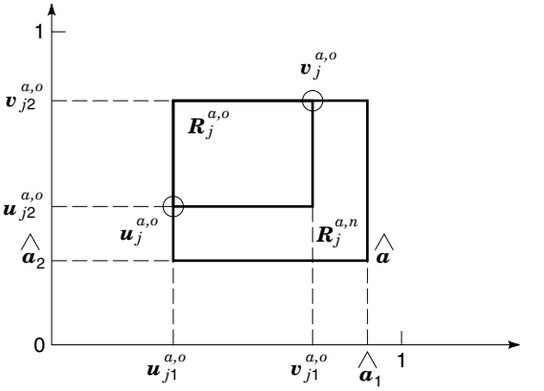

Figure 6. Input pattern $\hat{\boldsymbol{I}} = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}^c)$ represented by the point $\hat{\boldsymbol{a}}$, lies outside rectangle $R_j^{a,o}$ that represents template $\boldsymbol{w}_j^{a,o} = (\boldsymbol{u}_j^{a,o}, \{\boldsymbol{v}_j^{a,o}\}^c)$. Learning of $\hat{\boldsymbol{I}}$ creates a new rectangle $R_j^{a,n}$ (the rectangle including all the points of rectangle $R_j^{a,o}$ and the point $\hat{\boldsymbol{a}}$) of larger size than $R_j^{a,o}$.

It is worth pointing out that during the training process of Fuzzy ART or Fuzzy ARTMAP compressed representations of the input patterns, belonging to the training set, are formed at the F_2^a field. These compressed representations could be visualized as the rectangles corresponding to committed nodes in F_2^a. The idea of the rectangle corresponding to a node is that it includes within its boundaries all the input patterns that have been coded by this node. In Fuzzy ARTMAP, the compressed representations of the input patterns, formed in F_2^a, are mapped, during the training process, to appropriate output patterns (classes).

FUZZY ART EXAMPLE

The input patterns of the training list are given below. Furthermore, the Fuzzy network parameters are chosen as follows: $M_a = 2$, $\rho_a = 0.8$, $\alpha_a = 0.01$. Finally, the initial weights $\boldsymbol{w}_j^a$ are chosen equal to the "all-ones" vectors.

$$\begin{aligned}
\boldsymbol{I}^1 &= (0.20 \quad 0.20 \quad 0.80 \quad 0.80)\\
\boldsymbol{I}^2 &= (0.35 \quad 0.35 \quad 0.65 \quad 0.65)\\
\boldsymbol{I}^3 &= (0.30 \quad 0.50 \quad 0.70 \quad 0.50)\\
\boldsymbol{I}^4 &= (0.50 \quad 0.30 \quad 0.50 \quad 0.70)\\
\boldsymbol{I}^5 &= (0.32 \quad 0.32 \quad 0.68 \quad 0.68)\\
\boldsymbol{I}^6 &= (0.42 \quad 0.42 \quad 0.58 \quad 0.58)
\end{aligned} \tag{19}$$

First List Presentation

Present Pattern $\boldsymbol{I}^1$. Since no committed nodes exist in Fuzzy ART, node 1 in F_2^a will be activated and it will code input $\boldsymbol{I}^1$. After learning is over, the top-down vector from node 1 in F_2^a is equal to $\boldsymbol{w}_1 = \boldsymbol{I}^1$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^1$ in the first list, are pictorially shown in Fig. 7(a) (see R_1^a in the figure).

Present Pattern $\boldsymbol{I}^2$. The bottom-up inputs to nodes 1 and 2 in F_2^a are equal to 0.8457 and 0.4987, respectively. Node 1 will be activated first and it will pass the vigilance criterion, since $|\boldsymbol{I}^2 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^2| = 0.85 \geq \rho_a = 0.80$. After learning is over, $\boldsymbol{w}_1 = \boldsymbol{I}^1 \wedge \boldsymbol{I}^2 = (0.2 \quad 0.2 \quad 0.65 \quad 0.65)$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^2$ in the first list, are pictorially shown in Fig. 7(b) (see R_1^a in the figure).

Present Pattern $\boldsymbol{I}^3$. The bottom-up inputs to nodes 1 and 2 in F_2^a are equal to 0.9064 and 0.4987, respectively. Node 1 will be activated first and it will not pass the vigilance criterion, since $|\boldsymbol{I}^3 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^3| = 0.775 < \rho_a = 0.80$. Hence node 1 will be reset, and node 2 will be activated next. Node 2 will pass the vigilance criterion, since $|\boldsymbol{I}^3 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^3| = 1.0 \geq \rho_a = 0.80$. After learning is over, $\boldsymbol{w}_2 = \boldsymbol{I}^3 = (0.3 \quad 0.5 \quad 0.7 \quad 0.5)$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^3$ in the first list, are pictorially shown in Fig. 7(c) (see R_1^a and R_2^a in the figure).

Present Pattern $\boldsymbol{I}^4$. The bottom-up inputs to nodes 1, 2, and 3 in F_2^a are equal to 0.9064, 0.7960, and 0.4987, respectively. Node 1 will be activated first and it will not pass vigilance criterion, since $|\boldsymbol{I}^4 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^4| = 0.775 < \rho_a = 0.80$. Hence node 1 will be reset, and node 2 will be activated next. Node 2 will pass the vigilance criterion, since $|\boldsymbol{I}^4 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^4| = 0.8 \geq \rho_a = 0.80$. After learning is over, $\boldsymbol{w}_2 = \boldsymbol{I}^3 \wedge \boldsymbol{I}^4 = (0.3 \quad 0.3 \quad 0.5 \quad 0.5)$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^4$ in the first list, are pictorially shown in Fig. 7(d) (see R_1^a and R_2^a in the figure).

Present Pattern $\boldsymbol{I}^5$. The bottom-up inputs to nodes 1, 2, and 3 in F_2^a are equal to 0.9994, 0.9993, and 0.4987, respectively. Node 1 will be activated first and it will pass the vigilance criterion, since $|\boldsymbol{I}^5 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^5| = 0.85 \geq \rho_a = 0.80$. After learning is over, $\boldsymbol{w}_1 = \boldsymbol{I}^1 \wedge \boldsymbol{I}^2 \wedge \boldsymbol{I}^5 = (0.2 \quad 0.2 \quad 0.65 \quad 0.65)$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^5$ in the first list, are pictorially shown in Fig. 7(e) (see R_1^a, R_2^a, and R_3^a in the figure).

Present Pattern $\boldsymbol{I}^6$. The bottom-up inputs to nodes 1, 2, and 3 in F_2^a are equal to 0.9122, 0.9937, and 0.4987, respectively. Node 2 will be activated first and it will pass the vigilance criterion, since $|\boldsymbol{I}^6 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^6| = 0.80 \geq \rho_a = 0.80$. After learning is over, $\boldsymbol{w}_2 = \boldsymbol{I}^3 \wedge \boldsymbol{I}^4 \wedge \boldsymbol{I}^6 = (0.3 \quad 0.3 \quad 0.5 \quad 0.5)$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^6$ in the first list, are pictorially shown in Fig. 7(f) (see R_1^a, R_2^a, and R_3^a in the figure).

In the second list presentation $\boldsymbol{I}^1$, $\boldsymbol{I}^2$, $\boldsymbol{I}^3$, $\boldsymbol{I}^4$, $\boldsymbol{I}^5$, and $\boldsymbol{I}^6$ will be coded by $\boldsymbol{w}_1$, $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{w}_2$, $\boldsymbol{w}_1$, and $\boldsymbol{w}_2$, respectively. Also, in the second list presentation no weight changes will occur, and as a result we can declare the learning complete at the end of the first list presentation.

FUZZY ARTMAP EXAMPLE

The input patterns of the training list are given below. Furthermore, the Fuzzy ARTMAP network parameters are chosen as follows: $M_a = 2$, $\bar{\rho}_a = 0.8$, $\alpha_a = 0.01$. Finally, the initial weights $\boldsymbol{w}_j^a$ are chosen equal to the "all-ones" vectors.

$$\begin{aligned}
\boldsymbol{I}^1 &= (0.20 \quad 0.20 \quad 0.80 \quad 0.80)\\
\boldsymbol{I}^2 &= (0.35 \quad 0.35 \quad 0.65 \quad 0.65)\\
\boldsymbol{I}^3 &= (0.30 \quad 0.50 \quad 0.70 \quad 0.50)\\
\boldsymbol{I}^4 &= (0.50 \quad 0.30 \quad 0.50 \quad 0.70)\\
\boldsymbol{I}^5 &= (0.32 \quad 0.32 \quad 0.68 \quad 0.68)\\
\boldsymbol{I}^6 &= (0.42 \quad 0.42 \quad 0.58 \quad 0.58)
\end{aligned} \tag{20}$$

The corresponding output patterns are output pattern $\boldsymbol{O}^1$ for input patterns $\boldsymbol{I}^1$ and $\boldsymbol{I}^2$, output pattern $\boldsymbol{O}^2$ for input patterns $\boldsymbol{I}^3$ and $\boldsymbol{I}^4$, and output pattern $\boldsymbol{O}^3$ for input patterns $\boldsymbol{I}^5$ and $\boldsymbol{I}^6$.

First List Presentation

Present Pattern $\boldsymbol{I}^1$. Since no committed nodes exist in the F_2^a field of Fuzzy ARTMAP, node 1 in F_2^a will be activated and it will code input $\boldsymbol{I}^1$. After learning is over, the top-down vector from node 1 in F_2^a is equal to $\boldsymbol{w}_1 = \boldsymbol{I}^1$, and node 1 in F_2^a is mapped to output pattern $\boldsymbol{O}^1$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^1$ in the first list, are pictorially shown in Fig. 8(a) (see R_1^a in the figure). Rectangle R_1^a is mapped to output pattern $\boldsymbol{O}^1$

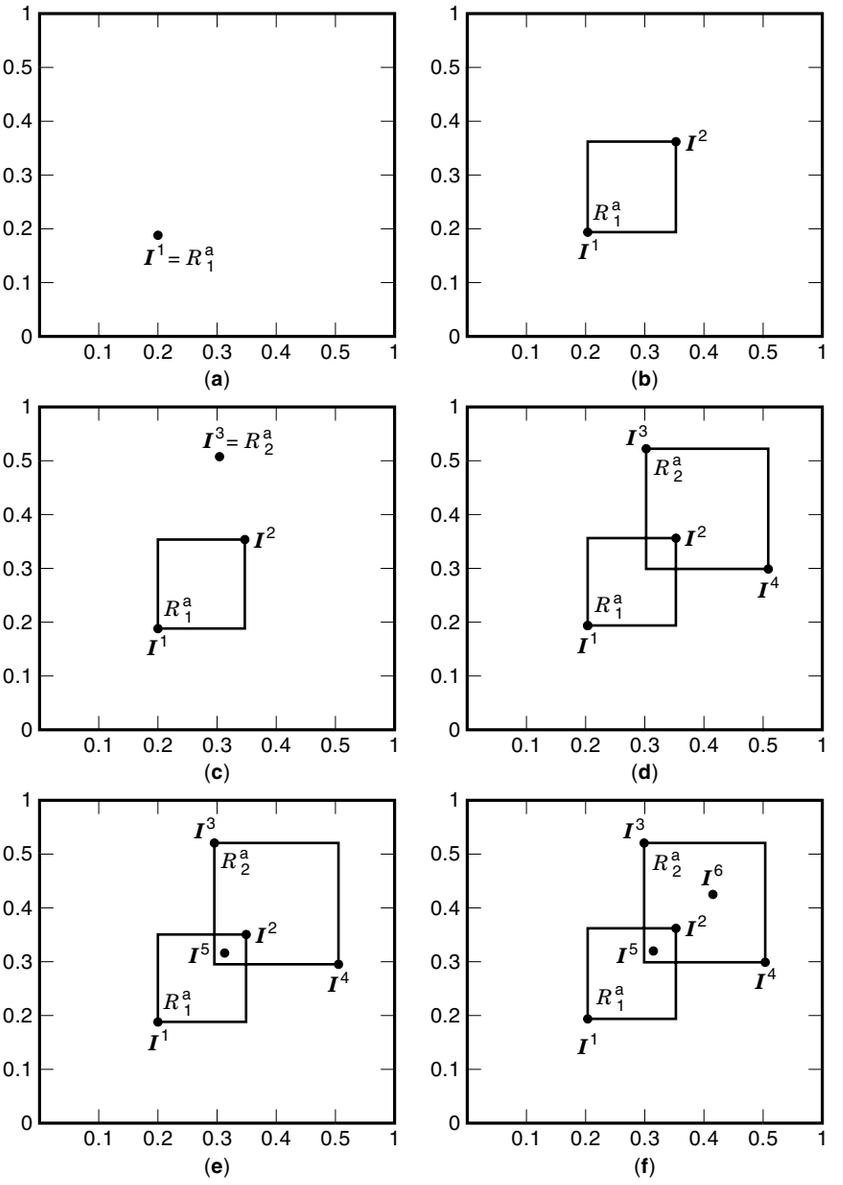

Figure 7. Rectangular representation of top-down templates in F_2^a during the first list presentation of the input patterns in the Fuzzy ART example.

Present Pattern $\boldsymbol{I}^2$. The bottom-up inputs to nodes 1 and 2 in F_2^a are equal to 0.8457 and 0.4987, respectively. Node 1 will be activated first and it will pass the vigilance criterion, since $|\boldsymbol{I}^2 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^2| = 0.85 \geq \rho_a = 0.80$. Also, node 1 in F_2^a is mapped to output pattern $\boldsymbol{O}^1$, which is the output pattern to which input pattern $\boldsymbol{I}^2$ needs to be mapped. Hence learning will take place, and after learning is over, $\boldsymbol{w}_1 = \boldsymbol{I}^1 \wedge \boldsymbol{I}^2 = (0.2 \quad 0.2 \quad 0.65 \quad 0.65)$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^2$ in the first list, are pictorially shown in Fig. 8(b) (see R_1^a in the figure). Rectangle R_1^a is mapped to output pattern $\boldsymbol{O}^1$.

Present Pattern $\boldsymbol{I}^3$. The bottom-up inputs to nodes 1 and 2 in F_2^a are equal to 0.9064 and 0.4987, respectively. Node 1 will be activated first and it will not pass the vigilance criterion, since $|\boldsymbol{I}^3 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^3| = 0.775 < \rho_a = 0.80$. Hence node 1 will be reset, and node 2 will be activated next. Node 2 will pass the vigilance criterion, since $|\boldsymbol{I}^3 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^3| = 1.0 \geq \rho_a = 0.80$. After learning is over, $\boldsymbol{w}_2 = \boldsymbol{I}^3 = (0.3 \quad 0.5 \quad 0.7 \quad 0.5)$, and node 2 is mapped to the output pattern $\boldsymbol{O}^2$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^2$ in the first list, are pictorially shown in Fig. 8(c) (see R_1^a and R_2^a in the figure). Rectangles R_1^a and R_2^a are mapped to output patterns $\boldsymbol{O}^1$ and $\boldsymbol{O}^2$, respectively.

Present Pattern $\boldsymbol{I}^4$. The bottom-up inputs to nodes 1, 2, and 3 in F_2^a are equal to 0.9064, 0.7960, and 0.4987, respectively. Node 1 will be activated first and it will not pass the vigilance criterion, since $|\boldsymbol{I}^4 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^4| = 0.775 < \rho_a = 0.80$. Hence node 1 will be reset, and node 2 will be activated next. Node 2 will pass the vigilance criterion, since $|\boldsymbol{I}^4 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^4| = 0.8 \geq \rho_a = 0.80$. Also, node 2 is mapped to the output pattern $\boldsymbol{O}^2$ to which the input pattern $\boldsymbol{I}^4$ needs to be mapped. Hence learning will occur, and after learning is over, $\boldsymbol{w}_2 = \boldsymbol{I}^3 \wedge \boldsymbol{I}^4 = (0.3 \quad 0.3 \quad 0.5 \quad 0.5)$. The committed top-down vectors in ART_a,

after the presentation of pattern $\boldsymbol{I}^4$ in the first list, are pictorially shown in Fig. 8(d) (see R_1^a and R_2^a in the figure). Rectangles R_1^a and R_2^a are mapped to output patterns $\boldsymbol{O}^1$ and $\boldsymbol{O}^2$, respectively.

Present Pattern $\boldsymbol{I}^5$. The bottom-up inputs to nodes 1, 2, and 3 in F_2^a are equal to 0.9994, 0.9993, and 0.4987, respectively. Node 1 will be activated first and it will pass the vigilance criterion, since $|\boldsymbol{I}^5 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^5| = 0.85 \geq \rho_a = 0.80$. But node 1 is mapped to the output pattern $\boldsymbol{O}^1$, while the input pattern $\boldsymbol{I}^5$ needs to be mapped to output pattern $\boldsymbol{O}^3$. Hence node 1 will be reset and the vigilance criterion in ART_a will be raised to a level slightly higher than $|\boldsymbol{I}^5 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^5| = 0.85$. Next, node 2 will be activated and node 2 will not pass the vigilance criterion, since $|\boldsymbol{I}^5 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^5| = 0.80 < \rho_a = 0.85^+$. Hence node 2 will be reset and node 3 will be activated next. Node 3 will pass the vigilance criterion, since $|\boldsymbol{I}^5 \wedge \boldsymbol{w}_3|/|\boldsymbol{I}^5| = 1.0 \geq \rho_a = 0.85^+$. After learning is over, $\boldsymbol{w}_3 = \boldsymbol{I}^5 = (0.32 \quad 0.32 \quad 0.68 \quad 0.68)$, and node 3 is mapped to the output pattern $\boldsymbol{O}^3$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^5$ in the first list, are pictorially shown in Fig. 8(e) (see R_1^a, R_2^a, and R_3^a in the figure). Rectangles R_1^a R_2^a, and R_3^a are mapped to output patterns $\boldsymbol{O}^1$, $\boldsymbol{O}^2$, and $\boldsymbol{O}^3$, respectively.

Present Pattern $\boldsymbol{I}^6$. The bottom-up inputs to nodes 1, 2, 3, and 4 in F_2^a are equal to 0.9122, 0.9993, 0.8955, and 0.4987, respectively. Node 2 will be activated first and it will pass the vigilance criterion, since $|\boldsymbol{I}^6 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^6| = 0.80 \geq \rho_a = 0.80$. But node 2 is mapped to the output pattern $\boldsymbol{O}^2$, while the input pattern $\boldsymbol{I}^6$ needs to be mapped to output pattern $\boldsymbol{O}^3$. Hence node 2 will be reset and the vigilance criterion in ART_a will be raised to a level slightly higher than $|\boldsymbol{I}^6 \wedge \boldsymbol{w}_2|/|\boldsymbol{I}^6| = 0.80$. Next, node 1 will be activated and node 1 will not pass the vigilance criterion, since $|\boldsymbol{I}^6 \wedge \boldsymbol{w}_1|/|\boldsymbol{I}^6|\rho_a = 0.80^+$. Hence node 1 will be reset and node 3 will be activated next.

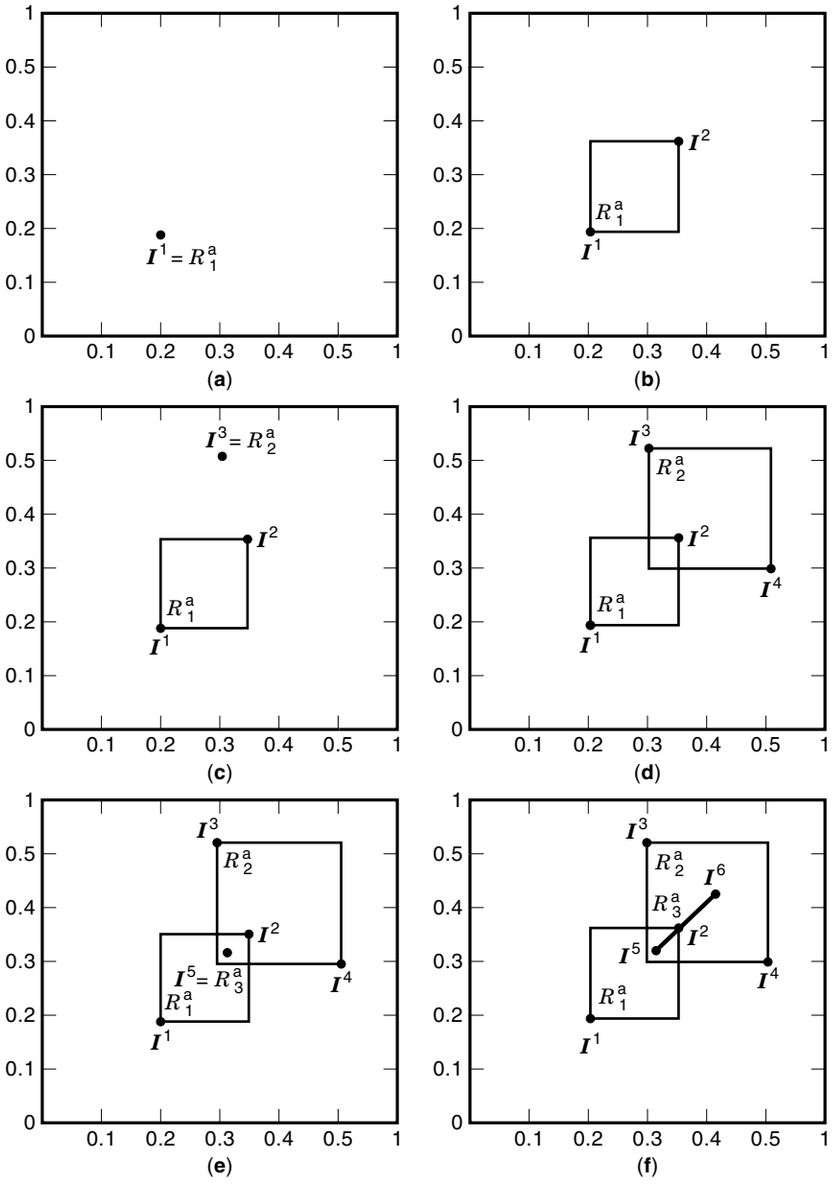

Figure 8. Rectangular representation of top-down templates in F_2^a during the first list presentation of the input/output pairs in the Fuzzy ARTMAP example.

Node 3 will pass the vigilance criterion, since $|\boldsymbol{I}^6 \wedge \boldsymbol{w}_3|/|\boldsymbol{I}| = 0.9 \geq \rho_a = 0.80^+$. Also, node 3 is mapped to the output pattern $\boldsymbol{O}^3$, which is the same output pattern to which the input pattern $\boldsymbol{I}^6$ needs to be mapped. Thus learning will take place, and after learning is over, $\boldsymbol{w}_3 = \boldsymbol{I}^5 \wedge \boldsymbol{I}^6 = (0.32 \quad 0.32 \quad 0.58 \quad 0.58)$. The committed top-down vectors in ART_a, after the presentation of pattern $\boldsymbol{I}^6$ in the first list, are pictorially shown in Fig. 8(f) (see R_1^a, R_2^a, and R_3^a in the figure). Rectangles R_1^a, R_2^a, and R_3^a are mapped to output patterns $\boldsymbol{O}^1$, $\boldsymbol{O}^2$, and $\boldsymbol{O}^3$, respectively.

In the second list presentation, $\boldsymbol{I}^1$, $\boldsymbol{I}^3$, $\boldsymbol{I}^4$, $\boldsymbol{I}^5$, and $\boldsymbol{I}^6$ will be coded by $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{w}_2$, $\boldsymbol{w}_3$, and $\boldsymbol{w}_3$, respectively. On the other hand, pattern $\boldsymbol{I}^2$ will be coded by a new node with template $\boldsymbol{w}_4 = \boldsymbol{I}^2$, and node 4 will be mapped to the output pattern $\boldsymbol{O}^1$. In the third presentation, patterns $\boldsymbol{I}^1$, $\boldsymbol{I}^2$, $\boldsymbol{I}^3$, $\boldsymbol{I}^4$, $\boldsymbol{I}^5$, and $\boldsymbol{I}^6$ will be coded by $\boldsymbol{w}_1$, $\boldsymbol{w}_4$, $\boldsymbol{w}_2$, $\boldsymbol{w}_2$, $\boldsymbol{w}_3$, and $\boldsymbol{w}_3$, respectively. Furthermore, all the input patterns are mapped to the correct output patterns, since nodes 1, 2, 3, and 4 are mapped to the output patterns $\boldsymbol{O}^1$, $\boldsymbol{O}^2$, $\boldsymbol{O}^3$, and $\boldsymbol{O}^1$, respectively. Also, in the third list presentation no weight changes will occur, and as a result we can declare the learning complete at the end of the second list presentation.

Note that in the Fuzzy ART and Fuzzy ARTMAP examples with two-dimensional data, the rectangles formed during learning can be of the trivial type {e.g., a point [R_1^a in Fig. 7(a)], or a line [R_3^a in Fig. 8(f)]}.

APPLICATIONS

The classification performance of Fuzzy ARTMAP has been examined against a plethora of pattern classification problems. In the original ARTMAP paper (5) the performance of the network with the mushroom database (25) was investigated. The mushroom database consists of 8124 input/output pairs of input/output features. The input features of the input vector represent each of the 22 observable features of a mushroom (e.g., cap-shape, gill-spacing, population, habitat). The output features of the output vector correspond to the mushroom classification in "edible" and "poisonous." Based on the results reported in Ref. 5, the Fuzzy ARTMAP system consistently achieved over 99% classification accuracy on the testing set with 1000 training input/output pairs; the testing set is the collection of input/output pairs (out of the 8124 possible) that were not included in the training set (1000 input/output pairs randomly chosen from the collection of 8124 input/output pairs). Classification accuracy of 95% was usually achieved with off-line training of 100–200 input/output pairs. The STAGGER algorithm (26), reached its maximum performance level of 95% accuracy after exposure to 1000 input/output training pairs. The HILLARY algorithm (27) demonstrated a performance similar to the STAGGER algorithm. Hence, for this database, Fuzzy ARTMAP was found to be an order of magnitude more efficient than the alternative systems.

Frey and Slate (28) developed a benchmark machine learning task that they describe as a "difficult categorization problem." The objective was to identify each of a large number of black-and-white rectangular pixel images as one of 26 capital letters A–Z. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a database of 20,000 characters. The fonts represent five different stroke styles (simplex, duplex, triplex, complex, and Gothic), and six different letter styles (block, script, italic, English, Italian, and German). Sixteen numerical feature attributes were then obtained from each character image, and each attribute value was scaled to a range of 0–15. The identification task was challenging because of the wide diversity among the different fonts and the primitive nature of the attributes.

Frey and Slate used this database to test the performance of a family of classifiers based on Holland's genetic algorithms. The training set consisted of 16,000 exemplars, with the remaining 4000 exemplars used for testing. Genetic algorithm classifiers having different input representations, weight update and rule creation schemes, and system parameters were systematically compared. Training was carried out for five epochs, plus a sixth "verification" pass during which no new rules were created, but a large number of unsatisfactory rules were discarded. In the Frey–Slate comparative study, these systems had correct classification rates that ranged from 24.5% to 80.2% on the 4000-item test set.

Fuzzy ARTMAP had an error rate on the letter recognition task that was consistently less than one-third that of the best Frey–Slate genetic algorithm classifiers. Of the 28 Fuzzy ARTMAP simulations reported in Ref. 6, the one with the best performance had a 94.7% correct prediction rate on the 4000-item test set, after five training epochs. Thus the error rate (5.3%) was less than one-third that of the best simulation in the Frey–Slate comparative study (19.2%).

Another paper (9) compared the performance of Fuzzy ARTMAP and its variants [ART-EMAP (7) and ARTMAP-IC (9)] with other algorithms, such as K-nearest neighbor (29), the ADAP perceptron (30), multisurface pattern separation (31), CLASSIT (32), instance-based (33), and C4 (34). The databases chosen for this comparison were the diabetes database, the breast cancer database, the heart disease database, and the gallbladder removal database (25). The basic conclusion out of this comparison is that Fuzzy ARTMAP, or its variants, performed as well or better than a variety of methods applied to the aforementioned benchmark problems.

In a recent publication Carpenter (35) produced a list of applications, where the family of ART networks and their variations have been used successfully. Below we reproduce this list with some additions of our own. A Boeing part retrieval system (36), satellite remote sensing (37,38), robot sensory-motor control (39–41), robot navigation (42), machine vision (43), three-dimensional object recognition (44), face recognition (45), image enhancement (46), Macintosh operating system software (47), automatic target recognition (48–50), electrocardiogram wave recognition (51,52), prediction of protein secondary structure (53), air quality monitoring (54), strength prediction for concrete mixes (55), signature verification (56), adaptive vector quantization (57), tool failure monitoring (58,59), chemical analysis from UV and IR spectra (60), frequency selective surface design for electromagnetic system devices (61), Chinese character recognition (62), and analysis of musical scores (63).

THEORETICAL RESULTS

In this section we investigate the learning properties of ART1, Fuzzy ART, and ARTMAP architectures. Some of the

learning properties discussed in this paper involve characteristics of the clusters formed in these architectures, while other learning properties concentrate on how fast it will take these architectures to converge to a solution for the type of problems that are capable of being solved. This latter issue is a very important issue in the neural network literature, and there are very few instances where it has been answered satisfactorily. It is worth noting that all of the results described in this section have been developed and proved elsewhere (2,4,16–18,20). In this article, we present these results in a unified manner, with the purpose of pointing out the similarities in the learning properties of the ART1, Fuzzy ART, and ARTMAP architectures.

Preliminaries

For the properties of learning in ART architectures, it is important to understand the distinctions among the top-down weights emanating from nodes in the F_2^a field. Consider an input $\boldsymbol{I}$ presented to the ART_a module. Consider also an arbitrary template of the ART_a module, designated as $\boldsymbol{w}^a$. A component of an input pattern $\boldsymbol{I}$ is indexed by i if it affects node i in the F_1^a field. Similarly, a component of a template $\boldsymbol{w}^a$ is indexed by i if it corresponds to the weight converging to node i in the F_1^a field. Based on this correspondence between the components of input patterns and templates in ART_a, we can identify three types of learned templates with respect to an input pattern $\boldsymbol{I}$: *subset templates*, *mixed templates*, and *superset templates*. A template $\boldsymbol{w}^a$ is a subset of pattern $\boldsymbol{I}$ if each one of the $\boldsymbol{w}^a$ components is smaller than or equal to its corresponding component in $\boldsymbol{I}$. A template $\boldsymbol{w}^a$ is a mixed template of pattern $\boldsymbol{I}$ if some of the $\boldsymbol{w}^a$ components are smaller than or equal to their corresponding components in $\boldsymbol{I}$, and the rest of the $\boldsymbol{w}^a$ components are larger than their corresponding components in $\boldsymbol{I}$. A template $\boldsymbol{w}^a$ is a superset of pattern $\boldsymbol{I}$ if each one of the $\boldsymbol{w}^a$ components is larger than or equal to its corresponding component in $\boldsymbol{I}$.

Besides the templates defined above, we also define an *uncommitted template* to be the vector of top-down weights associated with a node in F_2^a, which has not yet been chosen to represent an input pattern. As before, every component of an uncommitted template is equal to one. With reference to an input pattern $\boldsymbol{I}$, we also designate nodes in F_2^a as subset, mixed, superset, or uncommitted depending on whether their corresponding template is a subset, mixed, superset, or uncommitted template with respect to the input pattern $\boldsymbol{I}$.

One of the modeling assumptions required for the validity of some of the results presented in this section is fast learning. Fast learning implies that the input/output pairs presented to the ARTMAP architecture or the inputs presented to the ART1 and Fuzzy ART architectures are held at the network nodes long enough for the network weights to converge to their steady-state values. The learning equation for the weights provided by Eq. (5) is a learning equation pertaining to the fast learning scenario. Whenever the fast learning assumption is not imposed, we imply that the weights are modified in a slow learning mode; in the slow learning mode the input/output pairs (ARTMAP) or inputs (ART1, Fuzzy ART) are not applied at the network nodes long enough for the network weights to reach their steady-state values.

We have already defined before what we mean by the statement that "learning is complete" in the off-line training phases of the ART architectures. In the sequel, we provide this definition in more rigorous terms.

Definition 1. In the off-line training phase of an ART architecture the learning process is declared complete if every input pattern from the list chooses a node in the F_2^a field that satisfies the direct-access, no-learning conditions. A node j in F_2^a chosen by a pattern $\boldsymbol{I}$ satisfies the direct-access, no-learning conditions if (a) node j is the first node chosen in F_2^a by pattern $\boldsymbol{I}$, (b) node j is not reset, and (c) the top-down weights corresponding to node j (i.e., $\boldsymbol{w}_j^a$'s) are not modified. Conditions (a) and (b) are the direct-access conditions and condition (c) is the no-learning condition.

For example, in the case of an input list $(\boldsymbol{I}^1, \boldsymbol{O}^1)$, $(\boldsymbol{I}^2, \boldsymbol{O}^2)$, . . ., $(\boldsymbol{I}^P, \boldsymbol{O}^P)$, assume that list presentation n is the first list presentation at which each one of the input patterns chooses a node in F_2^a that satisfies the direct-access, no-learning conditions. In particular, assume that $\boldsymbol{I}^1$ chooses node j_1, $\boldsymbol{I}^2$ chooses node j_2, . . ., and $\boldsymbol{I}^P$ chooses node j_P, and nodes $j_1, j_2, \ldots, j_P$ satisfy the direct-access, no-learning conditions; the notation j_p $(1 \leq p \leq P)$ implies the node in F_2^a chosen by input pattern $\boldsymbol{I}^p$, and, as a result, we might have cases where $j_p = j_{p'}$ for $p \neq p'$. At the end of the nth list presentation we can declare that learning is complete. In the above example, no modification of the ART weights is performed during list presentation n. Hence we can further claim that learning is complete by the end of the $n-1$ list presentation. Obviously, in list presentations $\geq n$, input pattern $\boldsymbol{I}^1$ will always choose node j_1, input pattern $\boldsymbol{I}^2$ will always choose node j_2, and so on.

Properties of Learning

We will state a number of learning properties pertinent to the ART1, Fuzzy ART, and ARTMAP architectures. We will focus on learning properties that are common among the ART architectures under consideration.

Distinct Templates Property. The templates formed in ART1 with fast learning, Fuzzy ART, and ARTMAP with fast learning are distinct.

Direct Access by Perfectly Learned Template Property. In ART1, ARTMAP with fast learning, and Fuzzy ART, if an input pattern $\boldsymbol{I}$ has been perfectly learned by a node in the category representation field, this node will be directly accessed by the input pattern $\boldsymbol{I}$, whenever this input pattern is presented to the network architecture.

We say that an input pattern $\boldsymbol{I}$ has been perfectly learned by node j in F_2^a iff $\boldsymbol{w}_j^a = \boldsymbol{I}$.

Number of Templates Property. At the completion of the off-line training phase of ART1 and Fuzzy ART with fast learning, with enough nodes in the category representation layer, and small values for the network parameter α_a, the number of templates created is smaller than the number of patterns in the input list.

Order of Search Property. Suppose that $\boldsymbol{I}$ is an arbitrary pattern from the list of input patterns in ART1 and Fuzzy ART and from the list of input/output pairs in ARTMAP. Then, if

the network parameter value α_a is small, the largest subset template of $\boldsymbol{I}$ will be searched first. If this subset template is reset, all subset templates will be reset. If all learned subset templates are reset, then superset templates, mixed templates, and uncommitted templates are searched, not necessarily in that order.

Number of List Presentations Property—1. The off-line training phase of ART1, Fuzzy ART, and ARTMAP with fast learning, with enough nodes in the category representation field and small values of the network parameter α_a, will be complete in at most M_a list presentations.

Number of List Presentations Property–2. The off-line training phase of ART1 and Fuzzy ART with fast learning, with enough nodes in the category representation field and small values for the network parameter α_a, will be complete in m list presentations, where m represents the number of distinct size input patterns in the list.

The *Distinct Template Learning Property* is one of the good properties of the ART architectures. Since templates in ART1, Fuzzy ART, and ARTMAP represent compressed representations of the input patterns presented to these architectures, it would have been a waste to create templates that are equal.

The *Direct Access by a Perfectly Learned Property* is another indication that the ART architectures employ learning rules that behave in a sensible manner. This property is very essential for any pattern clustering or pattern classification machine. Since templates represent compressed representations of the input patterns presented to the architectures, we should expect an input pattern to point first to an equal (with the pattern) template as its most preferred representation versus any other template created by the architecture.

The *Number of Templates* property provides an upper bound for the number of nodes created in the category representation field of ART1 and Fuzzy ART so that these architectures can learn a list of input patterns, repeatedly presented to them. In practice, the number of templates created (or nodes required) in the category representation field is usually much less than the number of patterns in the input list and is an increasing function of the network parameters α_a and ρ_a.

The *Order of Search Property* is a very interesting result because it identifies the order according to which the templates created in the category representation field are accessed. This property is very instrumental in demonstrating the *Number of List Presentation Properties.*

The Number of List Presentation Properties of the ART architectures are somehow unique in the family of neural network architectures. To illustrate our point, consider the most popular back-prop network (64), where not only do we not know how many list presentations are required to learn an arbitrary list of input/output pairs but often we do not know whether the architecture will converge to a solution or not. On the contrary, for the ARTMAP architecture The *Number of List Presentations Property—1* tells us that we will need at most M_a list presentations to learn an arbitrary list of binary input/output pairs. The parameter M_a identifies the number of components of our input pattern $\boldsymbol{a}$, or the number of ones of our input pattern $\boldsymbol{I}$. This bound on the number of list presentations is a tight bound and it turns out to be very impressive if we consider a couple of examples. For instance, consider the case of input/output pairs, where the input patterns $\boldsymbol{a}$ have $M_a = 10$ (100) components; the input/output mapping problem that might be given to us in this example case can have at most $2^{10} \approx 1000$ ($2^{100} \approx 10^{30}$) input/output pairs. ARTMAP would need only at most 10 (100) presentations to learn this mapping problem. Can you imagine the time required by a back-prop network to learn a mapping problem involving a 10^{30} input/output pairs?

The *Number of List Presentation Property—2* tells us that the upper bound on the number of list presentations required by ART1 and Fuzzy ART to learn a list of input patterns, repeatedly presented to them, can get tighter. In particular, the number of list presentations required is upper bounded by the number of distinct size templates in the input list. For example, if $M_a = 100$ and the number of distinct size inputs presented to ART1 is 2, it will require 2 list presentations for ART1 to learn the list. This property is taken to extreme with Fuzzy ART, because in Fuzzy ART the preprocessing of the inputs leaves us with input patterns of the same size (M_a). Hence Fuzzy ART needs only one list presentation to learn the list of input patterns presented to it.

BIBLIOGRAPHY

1. S. Grossberg, Adaptive pattern recognition and universal recoding II: feedback, expectation, olfaction, and illusions. *Biol. Cybernet.,* **23**: 187–202, 1976.
2. G. A. Carpenter and S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comput. Vision, Graphics, Image Proc.,* **37**: 54–115, 1987.
3. G. A. Carpenter and S. Grossberg, ART 2: Self-organization of stable category recognition codes for analog input patterns. *Appl. Opt.,* **26** (23): 4919–4930, 1987.
4. G. A. Carpenter, S. Grossberg, and D. B. Rosen, Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks,* **4** (6): 759–771, 1991.
5. G. A. Carpenter, S. Grossberg, and J. H. Reynolds, ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks,* **4** (5): 565–588, 1991.
6. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Trans. Neural Networks,* **3** (5): 698–713, 1992.
7. G. A. Carpenter and W. D. Ross, ART-EMAP: A neural architecture for object recognition by evidence accumulation. *IEEE Trans. Neural Networks,* **6**: 805–818, 1995.
8. J. R. Williamson, Gaussian ARTMAP: A neural network for fast incremental learning of noisy multi-dimensional maps. *Neural Networks,* **9** (5): 881–897, 1996.
9. G. A. Carpenter and N. Markuzon, ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases. In *Technical Report CAS / CNS-96-017.* Boston: Boston University, 1996.
10. S. C. Newton, S. Pemmaraju, and S. Mitra, Adaptive fuzzy leader clustering model for pattern recognition. *IEEE Trans. Neural Networks,* **3** (5): 784–800, 1992.
11. M. J. Healy, T. P. Caudell, and S. D. G. Smith, A neural architecture for pattern sequence verification through inferencing. *IEEE Trans. Neural Networks,* **4** (1): 9–20, 1993.

12. Y. S. Kim and S. Mitra, An adaptive integrated fuzzy clustering model for pattern recognition. *Fuzzy Sets Syst.,* **65**: 297–310, 1994.
13. P. K. Simpson, Fuzzy Min-Max neural networks—part 1: Classification. *IEEE Trans. Neural Networks,* **3** (5): 776–786, 1992.
14. P. K. Simpson, Fuzzy Min-Max neural networks—part 2: Clustering. *IEEE Trans. Fuzzy Syst.,* **1** (1): 32–45, 1993.
15. C. A. Hung and S. F. Lin, Adaptive Hamming net: A fast learning ART1 model without searching. *Neural Networks,* **8**: 605–618, 1995.
16. M. Georgiopoulos, G. L. Heileman, and J. Huang, Convergence properties of learning in ART1. *Neural Comput.,* **2** (4): 502–509, 1990.
17. M. Georgiopoulos, G. L. Heileman, and J. Huang, Properties of learning related to pattern diversity in ART1. *Neural Networks,* **4** (6): 751–757, 1991.
18. M. Georgiopoulos, G. L. Heileman, and J. Huang, The N–N–N conjecture in ART1. *Neural Networks,* **5** (5): 745–753, 1992.
19. B. Moore, ART1 and pattern clustering. In D. S. Touretzky, G. Hinton, and T. Sejnowski (eds.), *Proceedings of the 1988 Connectionist Summer School.* San Mateo, CA: Morgan Kaufmann, 1989, pp. 175–185.
20. J. Huang, M. Georgiopoulos, and G. L. Heileman, Fuzzy ART properties. *Neural Networks,* **8** (2): 203–213, 1995.
21. M. Georgiopoulos et al., Order of search in Fuzzy ART and Fuzzy ARTMAP: A geometrical interpretation. In *Proceedings of the International Conference on Neural Networks,* Washington, DC: IEEE Press 1996, pp. 215–220.
22. G. Bartfai, On the match tracking anomaly of the ARTMAP neural network. *Neural Networks,* **9** (2): 295–308, 1996.
23. M. Georgiopoulos, J. Huang, and G. L. Heileman, Properties of learning in ARTMAP. *Neural Networks,* **7**: 495–506, 1994.
24. R. Dubes and A. Jain, Clustering techniques: The user's dilemma. *Pattern Recognition,* **8**: 247–260, 1976.
25. P. Murphy and D. Ada, UCI repository of machine learning databases. Technical report, Department of Computer Science, University of California, Irvine, CA, 1994.
26. J. S. Schlimmer, Concept acquisition through representational adjustment (technical report 87-19). Technical report, Doctoral Dissertation, Department of Information and Computer Science, University of California, Irvine, CA, 1987.
27. W. Iba, J. Wogulis, and P. Langley, Trading off simplicity and coverage in incremental concept learning. In *Proceedings of the 5th International Conference on Machine Learning.* Ann Arbor, MI: Morgan Kaufmann, 1988, pp. 73–79.
28. P. W. Frey and D. J. Slate, Letter recognition using Holland-style adaptive classifiers. *Mach. Learning,* **6**: 161–182, 1991.
29. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons, 1973.
30. J. W. Smith et al., Using the ADAP learning algorithm to forecast the onset of diabetes melitus. In *Proceedings Symposium on Computer Applications and Medical Care.* New York: IEEE Computer Society Press, 1988, pp. 261–265.
31. W. H. Wolberg and O. L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl. Acad. Sci. USA,* **87**: 9193–9196, 1990.
32. J. H. Gennari, P. Langley, and D. Fisher, Models of incremental concept formation. *Artif. Intell.,* **40**: 11–60, 1989.
33. D. W. Aha, D. Kibler, and M. K. Albert, Instance-based learning algorithms. *Mach. Learning,* **6**: 37–60, 1991.
34. J. R. Quinlan, The effect of noise on concept learning. In R. S. Michalski, J. C. Carbonell, and T. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach.* San Mateo, CA: Morgan Kaufmann, 1986, pp. 149–166.
35. G. A. Carpenter, Distributed learning, recognition, and prediction by ART and ARTMAP networks. In *Technical Report CAS / CNS-96-004.* Boston: Boston University, 1996.
36. T. P. Caudell et al., NIRS: Large scale ART-1 neural architectures for engineering design retrieval. *Neural Networks,* **7**: 1339–1350, 1994.
37. A. Baraldi and F. Parmiggiani, A neural network for unsupervised categorization of multivalued input patterns. *IEEE Trans. Geosci. Remote Sensing,* **33**: 305–316, 1995.
38. S. Gopal, D. M. Sklarew, and E. Lambin, Fuzzy-neural networks in multi-temporal classification of landcover change in sahel. In *Proceedings of DOSES Workshop on New Tools for Spatial Analysis.* Brussels, Luxemburg: DOSES, EUROSTAT, ECSC-EC-EAEC, 1994, pp. 55–68.
39. I. A. Bachelder, A. M. Waxman, and M. Seibert, A neural system for mobile robot visual place learning and recognition. In *Proceedings of World Congress on Neural Networks (WCNN93).* Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. I512–I517.
40. A. A. Baloch and A. M. Waxman, Visual learning, adaptive expectations, and learning behavioral conditioning of the mobil robot MAVIN. *Neural Networks,* **4** (3): 271–302, 1991.
41. A. Dubraski and J. L. Crowley, Learning locomotion reflexes: A self-supervised neural system for a mobile robot. *Robotics Autonomous Syst.,* 12: 133–142, 1994.
42. A. Dubraski and J. L. Crowley, Self-supervised neural system for reactive navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation,* Los Alamitos, CA, May 1994. New York: IEEE Computer Society Press, 1994, pp. 2076–2081.
43. T. P. Caudell and M. J. Healy, Adaptive Resonance Theory networks in the encephalon autonomous vision system. In *Proceedings of the 1994 IEEE International Conference on Neural Networks,* Piscataway, NJ. New York: IEEE Press, 1994, pp. II1235–II1240.
44. S. Seibert and A. M. Waxman, Adaptive 3D object recognition from multiple views. *IEEE Trans. Pattern Anal. Mach. Intell.,* **14**: 107–124, 1992.
45. S. Seibert and A. M. Waxman, An approach to face recognition using saliency maps and caricatures. In *Proceedings of the World Congress on Neural Networks (WCNN-93).* Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. III661–III664.
46. F. Y. Shih, J. Moh, and F. C. Chang, A new ART-based neural architecture for pattern classification and image enhancement without prior knowledge. *Pattern Recognition,* **25** (5): 533–542, 1992.
47. C. Johnson, Agent learns user's behavior. *Electr. Eng. Times,* 43–46, 1993.
48. A. M. Bernardon and J. E. Carrick, A neural system for automatic target learning and recognition applied to bare and camouflaged SAR targets. *Neural Networks,* **8**: 1103–1108, 1995.
49. M. W. Koch et al., Cueing, feature discovery, and one-class learning for synthetic aperture radar automatic target recognition. *Neural Networks,* **8**: 1081–1102, 1995.
50. A. M. Waxman et al., Neural processing of targets in visible, multispectral IR and SAR imagery. *Neural Networks,* **8**: 1029–1051, 1995.
51. F. M. Ham and S. W. Han, Quantitative study of the QRS complex using Fuzzy ARTMAP and the MIT/BIH arrhythmia database. In *Proceedings of the World Congress on Neural Networks (WCNN-93).* Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. I207–I211.

52. Y. Suzuki, Y. Abe, and K. Ono, Self-organizing QRS wave-recognition system in ECG using ART2. In *Proceedings of the World Congress on Neural Networks (WCNN-93).* Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. IV39–IV42.

53. B. V. Metha, L. Vij, and L. C. Rabelo, Prediction of secondary structures of proteins using Fuzzy ARTMAP. In *Proceedings of the World Congress on Neural Networks (WCNN-93).* Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. I228–I232.

54. D. Wienke, Y. Xie, and P. K. Hopke, An Adaptive Resonance Theory based artificial neural network (ART2-A) for rapid identification of airborne particle shapes from their scanning electron microscopy images. *Chemometrics and Intelligent Systems Laboratory,* 1994.

55. J. Kasperkiewicz, J. Racz, and A. Dubraswski, HPC strength prediction using artificial neural networks. *J. Comput. Civil Eng.,* **9:** 279–284, 1995.

56. N. A. Murshed, F. Bortozzi, and R. Sabourin, Off-line signature verification without a-priori knowledge of class w_2. A new approach. In *Proceedings of ICDAR 95: The Third International Conference on Document Analysis and Recognition,* 1995.

57. S. Mitra and S. Pemmaraju, Adaptive vector quantization using an ART-based neuro-fuzzy clustering algorithm. In *Proceedings of the International Conference on Neural Networks.* Washington, DC: IEEE Press, 1996, pp. 211–214.

58. S. Ly and J. J. Choi, Drill condition monitoring using ART-1. In *Proceedings of the 1994 IEEE International Conference on Neural Networks,* Piscataway, NJ. New York: IEEE Press, 1994, pp. II1226–II1229.

59. Y. S. Tarng, T. C. Li, and M. C. Chen, Tool failure monitoring for drilling purposes. In *Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing,* 1994, pp. 109–111.

60. D. Wienke and G. Kateman, Adaptive Resonance Theory based artificial neural networks for treatment of open-category problems in chemical pattern recognition—application to UV-Vis and IR spectroscopy. *Chemometrics and Intelligent Systems Laboratory,* 1994.

61. C. Christodoulou et al., Design of gratings and frequency selective surfaces using Fuzzy ARTMAP neural networks. *J. Electromagnet. Waves Appl.,* **9**: 17–36, 1995.

62. K. W. Gan and K. T. Lua, Chinese character classification using Adaptive Resonance network. *Pattern Recognition,* **25**: 877–888, 1992.

63. R. O. Gjerdingen, Categorization of musical patterns by self-organizing neuron-like networks. *Music Perception,* **7**: 339–370, 1990.

64. J. L. McClelland, D. E. Rumelhart, and G. E. Hinton, The appeal of parallel distributed processing. In D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* Cambridge, MA: MIT Press, 1986.

MICHAEL GEORGIOPOULOS
University of Central Florida

GREGORY L. HEILEMAN
University of New Mexico

JUXIN HUANG
Hewlett-Packard