Available online at www.sciencedirect.com
ScienceDirect
Neural Networks 20 (2007) 245–259
Neural Networks
www.elsevier.com/locate/neunet

Experiments with Safe μARTMAP: Effect of the network parameters on the network performance

Mingyu Zhong[a], Bryan Rosander[a], Michael Georgiopoulos[a,*,1], Georgios C. Anagnostopoulos[b], Mansooreh Mollaghasemi[c], Samuel Richie[a]

[a] *School of EECS, University of Central Florida, Orlando, FL 32816, United States*
[b] *Department of ECE, Florida Institute of Technology, Melbourne, FL 32901, United States*
[c] *Department of IEMS, University of Central Florida, Orlando, FL 32816, United States*

Received 6 October 2005; received in revised form 14 November 2006; accepted 14 November 2006

Abstract

Fuzzy ARTMAP (FAM) is currently considered to be one of the premier neural network architectures in solving classification problems. One of the limitations of Fuzzy ARTMAP that has been extensively reported in the literature is the category proliferation problem. That is, Fuzzy ARTMAP has the tendency of increasing its network size, as it is confronted with more and more data, especially if the data are of the noisy and/or overlapping nature. To remedy this problem a number of researchers have designed modifications to the training phase of Fuzzy ARTMAP that had the beneficial effect of reducing this category proliferation. One of these modified Fuzzy ARTMAP architectures was the one proposed by Gomez-Sanchez, and his colleagues, referred to as Safe μARTMAP. In this paper we present reasonable analytical arguments that demonstrate of how we should choose the range of some of the Safe μARTMAP network parameters. Through a combination of these analytical arguments and experimentation we were able to identify good default parameter values for some of the Safe μARTMAP network parameters. This feat would allow one to save computations when a good performing Safe μARTMAP network is needed to be identified for a new classification problem. Furthermore, we performed an exhaustive experimentation to find the best Safe μARTMAP network for a variety of problems (simulated and real problems), and we compared it with other best performing ART networks, including other ART networks that claim to resolve the category proliferation problem in Fuzzy ARTMAP. These experimental results allow one to make appropriate statements regarding the pair-wise comparison of a number of ART networks (including Safe μARTMAP).

Keywords: Machine learning; Classification; ARTMAP; Safe μARTMAP; Parameter settings; Entropy

1. Introduction

The Adaptive Resonance Theory (ART) was developed by Grossberg (1976). One of the most celebrated ART architectures is Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, & Reynolds, 1992), which has been successfully used in the literature for solving a variety of classification problems. Some of the advantages that Fuzzy ARTMAP possesses is that it can solve arbitrarily complex classification problems, it converges quickly to a solution (within a few presentations of the list of the input/output patterns belonging to the training set), it has the ability to recognize novelty in the input patterns presented to it, it can operate in an online fashion (new input/output patterns can be learned by the system without retraining with the old input/output patterns), and it produces answers that can be explained with relative ease.

Fuzzy ARTMAP has two limitations: (1) the order of the training examples greatly affects the performance of the network, and (2) the category proliferation problem is also extensively reported in the literature. The former limitation is alleviated by Safe μARTMAP (Gomez-Sanchez, Dimitriadis, Cano-Izquierdo, & Lopez-Coronado, 2001), and will be

* Corresponding author.
E-mail addresses: myzhong@ucf.edu (M. Zhong), bdrosander@gmail.com (B. Rosander), michaelg@mail.ucf.edu (M. Georgiopoulos), georgio@fit.edu (G.C. Anagnostopoulos), mollagha@mail.ucf.edu (M. Mollaghasemi), richie@mail.ucf.edu (S. Richie).
[1] Reprint requests to: Harris Center, Suite 345D, School of Electrical Engineering and Computer Science, University of Central Florida, 4000 Central Florida Blvd, Orlando, FL 32816, United States.

0893-6080/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2006.11.008

discussed in Section 2. Quite often the category proliferation problem, observed in Fuzzy ARTMAP architectures, is connected with the issue of overtraining in Fuzzy ARTMAP. Over-training happens when Fuzzy ARTMAP is trying to learn the training data perfectly at the expense of degraded generalization performance (i.e., classification accuracy on unseen data) and also at the expense of creating many categories to represent the training data (leading to the category proliferation problem). A number of authors have tried to address the category proliferation/overtraining problem in Fuzzy ARTMAP. Amongst them we refer to the work by Marriott and Harrison (1995), where the authors eliminate the match tracking mechanism of Fuzzy ARTMAP when dealing with noisy data, the work by Charalampidis, Kasparis, and Georgiopoulos (2001), where the Fuzzy ARTMAP equations are appropriately modified to compensate for noisy data, the work by Anagnostopoulos, Bharadwaj, Georgiopoulos, Verzi, and Heileman (2003), Anagnostopoulos, Georgiopoulos, Verzi, and Heileman (2002), Gomez-Sanchez, Dimitriadis, Cano-Izquierdo, and Lopez-Coronado (2002), Gomez-Sanchez et al. (2001), Verzi, Georgiopoulos, Heileman, and Healy (2001), where different ways are introduced of allowing the Fuzzy ARTMAP categories to encode patterns that are not necessarily mapped to the same label, the work by Koufakou, Georgiopoulos, Anagnostopoulos, and Kasparis (2001), where cross-validation is employed to avoid the overtraining/category proliferation problem in Fuzzy ARTMAP, and the work by Carpenter and Milenova (1998), Parrado-Hernandez, Gomez-Sanchez, and Dimitriadis (2003) and Williamson (1997), where the ART structure is changed from a winner-take-all to a distributed version and simultaneously slow learning is employed with the intent of creating fewer ART categories and reducing the effects of noisy patterns.

In this paper we focus our attention on one of these Fuzzy ARTMAP modifications, that is, Safe μARTMAP, introduce by Gomez-Sanchez et al. (2001). As reported in the literature, μARTMAP's approach to reduce the category proliferation problem is to allow categories in Fuzzy ARTMAP to encode input patterns that belong to different labels, thus eliminating the need of creating a new category every time an input pattern appears in the vicinity of categories of different labelling than the one that the input pattern possesses. Furthermore, μARTMAP allows some of the mixed label categories to be destroyed if they were too entropic (i.e. mixing of the labels within a category was too excessive). The μARTMAP network enforces the category destruction through a set of maximum allowed entropic thresholds. After a category is destroyed μARTMAP allows the creation of categories of smaller size, than the category that is destroyed. The performance (size of architecture created, and classification accuracy achieved on unseen data) by μARTMAP depends on the choice of the network parameters (i.e. entropic thresholds, ART baseline vigilance parameter, ART choice parameter, and order of training set pattern presentation to ART). The enhanced version of μARTMAP, called Safe μARTMAP, does not allow a category to expand too quickly, and avoids the creation of highly overlapped categories.

In this paper we contribute to the existing μARTMAP literature and in a bigger context to the ART literature by presenting reasonable analytical arguments that demonstrate how we could choose the range of the entropic thresholds in Safe μARTMAP. Furthermore, we perform an exhaustive experimentation to find the best Safe μARTMAP network for a variety of problems (simulated data and real data). The definition of "best" is dependent on the smallness of the network created, as well as the accuracy of the created network on a cross-validation accuracy (small networks of good accuracy are preferred). Through this experimentation we were able to define good default values for the Safe μARTMAP network parameters, applicable to a variety of problems. Finally, we identify the best performing Safe μARTMAP network (from the set of network parameter values that we have experimented with) and we compared it with other "best" performing ART networks, such as Fuzzy ARTMAP (Carpenter et al., 1992), Ellipsoidal ARTMAP (Anagnostopoulos & Georgiopoulos, 2001; Anagnostopoulos, 2001), Gaussian ARTMAP and Distributed Gaussian ARTMAP (Williamson, 1996), and their semi-supervised versions (Anagnostopoulos et al., 2003, 2002).

The rest of the paper is organized as follows. Section 2 provides a detailed description of μARTMAP and Safe μARTMAP. In Section 3 we discuss the Safe μARTMAP parameters in detail, and attempt to explain the effect of some of these parameter choices on the Safe μARTMAP performance. In Section 4 we provide experimental results. Some of the results in Section 4 pertain to the comparisons of Safe μARTMAP and other ART networks. The rest of the results in Section 4 pertain to the setting of default parameter values for Safe μARTMAP experimentations. Finally, in Section 5, we summarize our work and our conclusions from this work.

2. The μARTMAP architecture

In this section, we give a detailed description of μARTMAP. Some of the information included here cannot be found in the published μARTMAP references, and was obtained through private communication with Eduardo Gomez-Sanchez, one of the μARTMAP inventors.

The block diagram of the μARTMAP architecture is shown in Fig. 1. Note that the block diagram of Fig. 1 for μARTMAP is different from that of the μARTMAP architecture shown in the μARTMAP paper (see Gomez-Sanchez et al., 2002), and in the Fuzzy ARTMAP paper (Carpenter et al., 1992), but very similar to the block diagram shown in Kasuba (1993), and Taghi, Bagmisheh, and Pavesic (2003). In Kasuba and Taghi, a simplified version of the supervised ART architectures was introduced that is equivalent to the more complicated architecture depicted in Carpenter et al. (1992) and Gomez-Sanchez et al. (2002), but much simpler to understand. The equivalence stated in the previous statement is valid only for classification problems.

The μARTMAP architecture of the block diagram of Fig. 1 has three major layers. The input layer (F_1^a) where the input patterns (designated by $\mathbf{I}$) are presented, the category representation layer (F_2^a) where compressed representations of

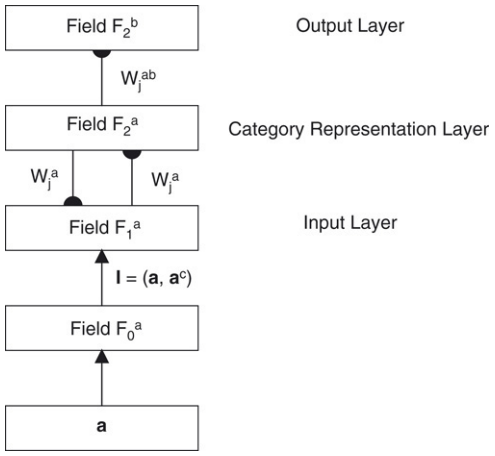

Fig. 1. Block-diagram of μARTMAP architecture.

these input patterns (designated as $\mathbf{w}_j^a$ and called templates) are formed, and the output layer (F_2^b) that holds the labels of the categories formed in the category representation layer. Another layer shown in Fig. 1, and designated by F_0^a, is a pre-processing layer and its functionality is to pre-process the input patterns before their presentation to μARTMAP. The pre-processing operation, called complementary coding, takes an input pattern $\mathbf{a}$ and expands it by appending to it the complement of a, designated as $\mathbf{a}^c$. That is, the input pattern $\mathbf{I}$ to μARTMAP is now equal to

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) \tag{1}$$

where $\mathbf{a}^c = \mathbf{1} - \mathbf{a}$. It is assumed here that every component of the original vectors $\mathbf{a}$ is normalized into the interval $[e, 1 - e]$ (where e is a small positive number, as explained in Section 3). The number of nodes in the input layer, the category representation layer, and the output layer of μARTMAP is designated by $2M_a$ (M_a is the dimensionality of the vector $\mathbf{a}$), N_a, and N_b (number of classes), respectively. Note that M_a and N_b are deterministic while N_a is changing dynamically in the training process.

There are a number of weights in the μARTMAP architecture that are worth mentioning: (a) the vector of weights (templates) emanating from every node in the category representation layer and converging to all the nodes in the input layer (same as in Fuzzy ARTMAP; the vector of weights emanating from node j in the category representation layer and converging to all the nodes in the input layer is designated by $\mathbf{w}_j^a = (w_{j1}^a, \ldots, w_{ji}^a, \ldots, w_{j\,2M_a}^a)$); (b) the vector of weights emanating from every node in the category representation layer and converging to all the nodes in the output layer. The vector of weights emanating from node j in the category representation layer and converging to all the nodes in the output layer is designated by $\mathbf{W}_j^{ab} = (W_{j1}^{ab}, \ldots, W_{jk}^{ab}, \ldots, W_{j\,N_b}^{ab})$, and component W_{jk}^{ab} of this weight vector represents the number of times that node j has been chosen by an input pattern. Note that the vector weights, designated by $\mathbf{w}_j^a$, have an interesting geometrical interpretation. Its first M_a components define the lower endpoint of a hyper-box, while its last M_a components define the upper endpoint of the hyper-box. It is assumed throughout this paper that the reader is familiar with the geometrical (hyper-box) interpretation of the template weights in ART.

μARTMAP operates in two phases: The training phase and the performance phase. In the training phase of μARTMAP we have a collection of input/associated labels pairs (called training set), and we present it to μARTMAP, one input/associated label pair $\{\mathbf{I}, \mathit{label}(\mathbf{I})\}$ at a time, in a manner that will be further explained below. The performance phase of μARTMAP will be explained after the training phase has been discussed.

The training phase of μARTMAP is succinctly described as follows (Steps 1–2):

(1) (Learning Phase) Find the nearest category in the category representation layer of μARTMAP that resonates with the input patterns.
   (a) If the label of the input pattern passes both the resonance test and the entropy test (see the context) update the weights of this category.
   (b) Otherwise, reset the winner, and try the next winner. Uncommitted nodes (categories) are chosen if and only if we cannot find a winner node from the list of already committed nodes.

(2) (Offline Evaluation Phase) After the learning phase is finished (i.e. all input/associated label pairs of the training set have chosen a committed node) we present all the patterns again to check the total entropy of the created categories, without changing any $\mathbf{w}_j^a$ vector. One pass of the learning phase and the offline evaluation phase is called one *epoch*.
   (a) If the total entropy is below a designated threshold, or if the number of epochs reaches a preset maximum value, training is completed.
   (b) If not, the category that contributes the most to the total entropy value is destroyed (e.g. in Fig. 2, the darkest rectangle will be removed if the training continues), the vigilance threshold in μARTMAP is increased to be slightly higher than the vigilance level of the category destroyed, and the next epoch will be started. In the learning phase of the next epoch, however, we present to μARTMAP only the training patterns that chose the destroyed category in the learning phase (rather than offline evaluation phase) of this epoch or the previous epochs. In the offline evaluation of the next epoch, we still present all the patterns.

The nearest category (mentioned in Step 1) to an input pattern $\mathbf{I}$ presented to μARTMAP is determined by finding the category that maximizes the function:

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

where $|\mathbf{x}|$ is the size of the vector $\mathbf{x}$ and it is defined to be the sum of its components.

The above function is called the bottom-up input (or choice function) pertaining to the F_2^a layer node j with category representation (template) equal to the vector $\mathbf{w}_j^a$, due to the presentation of input pattern $\mathbf{I}$. This function obviously depends

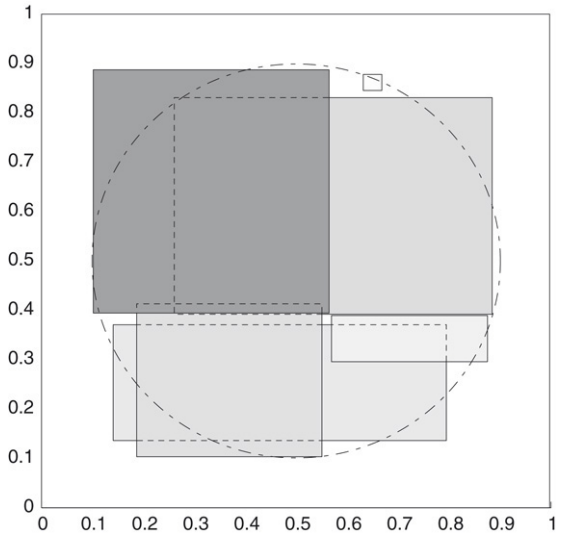

Fig. 2. μARTMAP Training results after 20 epochs on the circle-in-square database. In the database, 1000 points are uniformly distributed in the square. The points inside the circle (the circle has the same center as the square, and its area is half of the area of the square) are labelled as class 1 and the points outside the circle are labelled as class 2. Each rectangle, including the outermost rectangle whose width and height are both very close to one, represents a category in the network. The face colour of the rectangle represents the category entropy in the offline evaluation (h_j^{off}) at the end of the 20th epochs (a darker face colour implies a higher entropy value for the data inside the rectangle).

on the μARTMAP parameter α, called choice parameter, which assumes values in the interval $(0, \infty)$. In most simulations of ART architectures the useful range of α is the interval (0, 10). The resonance of a category (also mentioned in Step 1) is determined by the resonance test, which examines if the vigilance ratio, defined below

$$\rho(\mathbf{I}, \mathbf{w}_j^a) = \frac{|\mathbf{I} \wedge \mathbf{w}_j^a|}{M_a} \tag{3}$$

satisfies the following condition:

$$\rho(\mathbf{I}, \mathbf{w}_j^a) \geq \rho_a \tag{4}$$

where the fuzzy min operation applied on two vectors $\mathbf{x}$ and $\mathbf{y}$, designated by $\mathbf{x} \wedge \mathbf{y}$, is a vector whose components are equal to the minimum of the corresponding components of $\mathbf{x}$ and $\mathbf{y}$.

If the above equation is satisfied we say that resonance is achieved. The parameter ρ_a appearing in the above inequality is called the vigilance parameter and assumes values in the interval [0, 1]. For Fuzzy ARTMAP, this parameter is globally applied to all categories. In μARTMAP, however, each category has its own ρ_a parameter, which is initialized as the global vigilance level when the category is committed and will not be changed afterwards; the global vigilance level is never used in the vigilance test, and it will not affect the existing categories' ρ_a parameters even when it is raised according to (10). At the beginning of training this parameter is set equal to a baseline vigilance level, designated by $\bar{\rho}_a$, which assumes values in the interval [0, 1], and is set by the user. After training commences the vigilance level is allowed to change and become larger than the baseline vigilance level, as categories in μARTMAP are destroyed for being too entropic (see Step 2(b) of the algorithm). Increased values of the vigilance level produce more nodes in the category representation layer of μARTMAP. If a chosen category j in μARTMAP passes the resonance test then this category is allowed to encode the presented input/associated label pair in the following manner:

$$\mathbf{w}_j^a = \mathbf{w}_j^a \wedge \mathbf{I}, \qquad W_{jk}^{ab} = W_{jk}^{ab} + 1 \tag{5}$$

where $k = label(\mathbf{I})$. The update of the templates, illustrated by the above equation, has been called *fast-learning* in the ART literature.

If the category j is chosen and it resonates but it fails the entropy test, that is, the entropy of this category is higher than the allowable entropic threshold of a category, then this category is reset, the changes in (5) are undone, and the algorithm goes back to examine the rest of the categories to identify a new winner that resonates. Note that in this case the vigilance level is not increased and this is one of the differences between μARTMAP and the Fuzzy ARTMAP algorithms. The entropy of a category is defined by the following equation:

$$\begin{aligned} h_j &= \frac{|\mathbf{W}_j^{ab}|}{|\mathbf{W}^{ab}|}\text{entropy}(j) \\ &= -\frac{|\mathbf{W}_j^{ab}|}{|\mathbf{W}^{ab}|}\sum_{k=1}^{N_b}\frac{\mathbf{W}_{jk}^{ab}}{|\mathbf{W}_j^{ab}|}\log_2\left(\frac{W_{jk}^{ab}}{|\mathbf{W}_j^{ab}|}\right) \end{aligned} \tag{6}$$

where

$$|\mathbf{W}_j^{ab}| = \sum_{k=1}^{N_b} W_{jk}^{ab}, \qquad |\mathbf{W}^{ab}| = \sum_{j=1}^{N_a}\sum_{k=1}^{N_b} W_{jk}^{ab}. \tag{7}$$

In the above equations entropy(j) measures the entropy of node j, and h_j is the value of this entropy weighted by the relative frequency with which this node has encoded input/associated label pairs before. The level of this weighted entropy of the node that μARTMAP allows is a μARTMAP parameter value, denoted as $h_{\max}$, and it is a value that has to be defined by the user. By using the entropy test, μARTMAP allows $\bar{\rho}_a$ to be set small and large categories to be created as long as they are pure enough. This mechanism makes it possible to produce a network with very few categories and sufficient accuracy.

In Safe μARTMAP, the winner category must also pass a distance test if it is already committed:

$$\frac{|\mathbf{w}_j^a| - |\mathbf{I} \wedge \mathbf{w}_j^a|}{M_a} \leq \delta \tag{8}$$

where δ is also specified by the user, taking value from $(0, 1 - \bar{\rho}_a]$. This test requires that the size change of the winner category should not be too large due to a single pattern update. If the winner category fails this test, no other categories will be picked to learn this pattern at this point. Instead, this pattern remains "unlearned". After all patterns are presented (which is called a *pass*), the unlearned patterns are presented again in the next *pass*. This time the previous winner categories may learn these patterns. If no pattern is learned in a whole pass, an unlearned pattern will be picked and a new category

will be committed to learn this pattern; then all the other unlearned patterns are presented in the next pass. The above is repeated until all patterns are learned. In this way, the learning phase of a single epoch may consist of many passes. This is the only difference between Safe μARTMAP and the original μARTMAP.

After the 1st epoch of training is completed we examine the total entropy of the categories created. To do so we feed all the training input patterns to the trained μARTMAP architecture and we keep the count of how many times category j has been chosen by a pattern in the training set whose corresponding label is label k. In this cycle (referred to as offline evaluation phase) a pattern chooses the category that receives the highest bottom-up input. At the end of this processing cycle we calculate a matrix $\mathbf{V}^{ab} = [V_{jk}^{ab}]$, where V_{jk}^{ab} is the number of patterns in class k that selects category j in the offline evaluation. Although $\mathbf{V}^{ab}$ is defined similarly as $\mathbf{W}^{ab}$, they usually have different values because in the offline evaluation, the input patterns might choose a different category than in the one chosen in the learning phase, since new categories have been created. The total entropy of the categories in μARTMAP is now defined as:

$$H = \sum_{j=1}^{N_a} h_j^{\text{off}} = \sum_{j=1}^{N_a} -\frac{|\mathbf{V}_j^{ab}|}{|\mathbf{V}^{ab}|} \sum_{k=1}^{N_b} \frac{V_{jk}^{ab}}{|\mathbf{V}_j^{ab}|} \log_2 \left(\frac{V_{jk}^{ab}}{|\mathbf{V}_j^{ab}|} \right). \tag{9}$$

If the total entropy H is larger than a pre-specified (by the user) threshold, designated as $H_{\max}$, then the total entropy is considered to be too high and the category j with the largest h_j^{off} is chosen, and destroyed. Then, the patterns that chose this category as their representative category in the learning phase of one of the previous epochs are presented again to μARTMAP. The global vigilance parameter level is increased to a value slightly higher than the vigilance ratio of the destroyed category:

$$\rho_a = \min \left(1, \frac{|\mathbf{w}_j^a|}{M_a} + \Delta\rho \right) \tag{10}$$

where the parameter $\Delta\rho$ is chosen to be a small positive constant. As mentioned before, this new vigilance level will affect only the categories created in the next epoch (so that the destroyed category cannot be created again); the existing categories will keep their ρ_a values.

In the performance phase of μARTMAP, a test input is presented to the input layer of μARTMAP and the node in the category representation layer that receives the maximum bottom-up input is chosen (say node j). No resonance test or entropy test is performed. Then the predicted label for this test input is chosen to be the label that most often node j has been mapped to in μARTMAP's training process. That is the predicted label of this input is chosen to be the label k that maximizes W_{jk}^{ab}.

3. Discussion of Safe μARTMAP parameters

As we have mentioned above, Safe μARTMAP introduces four new network parameters $(e, h_{\max}, H_{\max}, \delta)$ that are different than the typical set of ART parameters used in ART networks. In the sequel, we elaborate on each one of these new Safe μARTMAP's parameters in an effort to understand their functionality better. Furthermore, this analysis will help us define the range of the parameter values $h_{\max}$, $H_{\max}$ that someone needs to experiment with in order to find the best performing Safe μARTMAP network for a specific classification problem. At the end of this section we also revisit the typical ART parameters (i.e., α, $\bar{\rho}_a$, $\Delta\rho$) and we state what values for these parameters we have used in our Safe μARTMAP experiments.

3.1. Discussion of parameter e

The choice parameter α in μARTMAP affects the competition of the nodes, according to (2). It is desired that:

(1) if a point (representing an input pattern) is inside two hyper-boxes (whose boundaries are defined by the corresponding categories in the network), it should choose the smaller hyper-box;
(2) if a point (representing an input pattern) is inside one hyper-box, no matter how large the hyper-box is, and outside another hyper-box, no matter how small this hyper-box is, it should choose the former one.

Condition (1) simply requires $\alpha > 0$. Condition (2) cannot be satisfied if $|\mathbf{w}_j^a|$ can be arbitrarily small (or the box can be arbitrarily large). For the databases with input elements normalized to [0, 1], no positive α value allows a box to cover the whole input space (which means $|\mathbf{w}_j^a| = |\mathbf{I} \wedge \mathbf{w}_j^a| = 0$) and satisfy Condition (2) at the same time. The authors of μARTMAP (personal communication with Gomez-Sanchez) accounted for that by choosing the elements of every input pattern, presented to μARTMAP, to lie in the interval $[e, 1-e]$ instead of [0, 1]. In particular, μARTMAP algorithm requires that:

(1)
$$\alpha \ll \min |\mathbf{w}_j^a| = 2M_a e \tag{11}$$
so that when a point is inside a hyper-box (represented by the weight vector $\mathbf{w}_j^a$), the corresponding T_j is close to one even if the hyper-box covers the whole input space.

(2)
$$e \ll 1 \tag{12}$$
or otherwise the vigilance test would always pass when the vigilance parameter ρ_a is small, since
$$\frac{|\mathbf{I} \wedge \mathbf{w}_j^a|}{|\mathbf{I}|} \geq \frac{2M_a e}{M_a} = 2e.$$

It is interesting to observe that the introduction of e is equivalent to redefining the fuzzy sum operand. Assume $\mathbf{I}$ is a pattern in the database whose attributes are normalized to $[e, 1-e]$, and let $\bar{\mathbf{I}}$ denote the same pattern in the database normalized to [0, 1]. It can be easily shown that:

$$\mathbf{I} = (1-2e)\bar{\mathbf{I}} + [e, e, e, \ldots, e]$$
$$|\mathbf{I}| = (1-2e)|\bar{\mathbf{I}}| + 2M_a e$$
$$|\mathbf{I}_1 \wedge \mathbf{I}_2| = (1-2e)|\bar{\mathbf{I}}_1 \wedge \bar{\mathbf{I}}_2| + 2M_a e.$$

Therefore, if we redefine the size operand, as articulated in the above equations, for calculations pertinent to the category representation layer (such as calculation of bottom-up inputs, calculation of vigilance ratios, update of the template values), there is no need of renormalizing the attributes to $[e, 1-e]$ as long as they are already normalized to [0, 1]. This redefinition of the size operand does not apply though to the calculation of the size of the inter-ART weights $\mathbf{W}_j^{ab}$, $\mathbf{V}_j^{ab}$ (i.e., $|\mathbf{W}_j^{ab}|$ or $|\mathbf{V}_j^{ab}|$); the old definition of the size operand is applicable to these weights.

In our Safe μARTMAP experiments, the choice parameter α is set to 0.01 and 0.001. Furthermore, the minimum M_a value is equal to 2. Due to the above constraints, it turns out that $1/400 \ll e \ll 1$. Hence, we set e to 0.05 in our μARTMAP experiments. Our experiments have also shown (not elaborated in this paper) that the μARTMAP network performance is not sensitive to the values of α or e, as long as the constraints (11) and (12) are satisfied.

3.2. Discussion of parameter $H_{\max}$

The parameter $H_{\max}$ controls the impurity of the entire μARTMAP network. It terminates the training process to prevent over-training (its primary function is to prevent over-training). $H_{\max}$ has a direct effect on the final accuracy of the μARTMAP network. Setting $H_{\max} = 0$ means that the μARTMAP must have 100% accuracy on the training set in the offline evaluation, which is usually impractical, and it does not necessarily improve μARTMAP's performance on unseen data. In most cases, setting $H_{\max} = 0$ not only keeps the training algorithm running for a long time (introduces extra computational cost in the training phase), but also overfits the network to the training set, which results in category proliferation and reduction of the network's predictive accuracy on unseen data (generalization). On the other hand, setting $H_{\max}$ to a very high value will terminate the training iteration too soon and result in low accuracy of predicting the labels of the training set as well as the labels of unseen data.

Apparently, the proper $H_{\max}$ value is problem-dependent. Nevertheless, in the sequel we discuss a number of ways of producing estimates for $H_{\max}$.

First, let N_b denote the number of classes (namely the number of nodes in the output layer), and $\hat{A}$ represent the expected accuracy given by the user and assumed to lie in the interval $(1/N_b, 1]$. If the theoretical optimal accuracy of a problem is known, then $\hat{A}$ is equal to this optimal accuracy. Of course, $\hat{A}$ is, quite often, unknown. However, guessing $\hat{A}$ (e.g., by using the information of how well other classifiers have performed on this problem) is easier than guessing $H_{\max}$. When the accuracy of the network on the training set reaches $\hat{A}$, the network tends to have the best accuracy on unseen patterns, as long as the training parameters are set properly (see Appendix A). In Appendix A we demonstrate how we can produce lower and upper bounds for the $H_{\max}$ value, given an estimate $\hat{A}$ of this theoretical optimal accuracy that a classifier can attain. These lower and upper bounds of $H_{\max}$ are shown in the following equations.

$$\begin{aligned} &H_L \le H \le H_U \\ &H_L = -\log_2 \hat{A} \\ &H_U = -\hat{A}\log_2 \hat{A} - (1-\hat{A})\log_2 \frac{1-\hat{A}}{N_b - 1}. \end{aligned} \tag{13}$$

In practice, both H_L and H_U can be far away from a good $H_{\max}$ value. In Appendix A, we provide examples of classification problems that justify two other estimates (in addition to H_L and H_U) of a good value for $H_{\max}$. These estimates are provided below, and denoted as H_{E1}, and H_{E2}.

$$\begin{aligned} H_{E1} &= \frac{(1-\hat{A})N_b \log_2 N_b}{N_b - 1} \\ H_{E2} &= -\frac{N_b(1-\hat{A}) - (N_b - 1)p}{1-p}\log_2 p - \log_2 \hat{A} \end{aligned} \tag{14}$$

where $p \in [0, 1)$ is the solution to the equation $\frac{1-p}{1-p^{N_b}} = \hat{A}$. In Section 4 we use these aforementioned entropy calculations to produce specific parameter settings for $H_{\max}$ used in our experimentations with μARTMAP.

3.3. Discussion of parameter $h_{\max}$

Parameter $h_{\max}$ controls the impurity of each node in the μARTMAP network. Setting $h_{\max} = 0$ means that all the nodes must be completely pure when created or expanded which suffers from the same shortcomings as choosing $H_{\max} = 0$ that have been discussed above. Setting $h_{\max} = \infty$ means the entropy test always passes, which implies that μARTMAP creates clusters of patterns without any regard for their respective label.

Parameter $h_{\max}$ affects the training process mostly in the first epoch. Beyond the first epoch, only the patterns learned by the category that is removed in the previous epoch will be presented. If these patterns are learned in exactly the same order as in the last time, then the entropy test will always pass no matter whether or not they go to a new category. This is because when each one of these patterns is learned again, the total number of learned patterns $|\mathbf{W}^{ab}|$ is no less than what it was the last time the pattern was learned; also $|\mathbf{W}_j^{ab}|\text{entropy}(j)$ will never increase when a pattern is removed from category j, due to the concavity of the entropy function. However, if the learning order is changed, due to the increased vigilance level, the entropy test may fail, as shown in the next paragraph.

According to (6), it is difficult to estimate a good $h_{\max}$ value, since $|\mathbf{W}_j^{ab}|$, the number of patterns learned by category j, is not easy to predict. Moreover, h_j is much more sensitive to the order in which the patterns are presented than the total entropy is during the offline evaluation phase. For example, suppose there is only one category in the network, and the first four patterns that μARTMAP learned have the class labels 1, 1, 1, 2, respectively (as in the order of the list presentation). The h_j values would be 0, 0, 0, 0.8113, after category j learns these patterns. If we swap the second and the fourth patterns, then the h_j values would be 0, 1, 0.9183, 0.8113, after category j learns

these patterns. If we set $h_{\max} = 0.9$, then category j would learn all the four patterns in the first case (before swapping), but it would not learn the pattern with class label 2 in the second case (after swapping).

In our experiments, we take the proper value of $h_{\max}$ to be proportional to the proper value of $H_{\max}$ (if we assume that all nodes have the same number of patterns, and that the patterns always select the same node during the offline evaluation as they do during the learning phase, then we can set $h_{\max}$ to be equal to $H_{\max}/N_a$). Since the optimal $h_{\max}/H_{\max}$ ratio might be problem-dependent, it is very difficult to estimate this ratio as we do for $H_{\max}$. In our experiments, we vary the ratio $h_{\max}/H_{\max}$ in order to identify the best such ratio.

3.4. Discussion of parameter δ

Parameter δ controls the *size change per pattern* of each category, according to (8). This parameter alleviates the overlapping problem in μARTMAP and reduces the effect of the network's dependence on the order of training pattern presentation. Small δ values mean that every time there is a category size change in Safe μARTMAP this size change must be small. Quite frequently, a small δ value will cause longer training time because in each epoch, many patterns will be placed back into the unlearned set for a number of consecutive epochs. If $\delta = 0$, then no category can increase its size, which is equivalent to setting $\bar{\rho}_a = 1$. If $\delta \geq 1 - \bar{\rho}_a$, then (8) is always satisfied, and Safe μARTMAP reduces to μARTMAP. The optimal δ value is also dependent on the distribution of patterns. Although δ makes the algorithm less sensitive to the pattern list order, the optimal value of δ depends on the distribution of the data points more than the other network parameters do. The parameter δ also depends on the number of training patterns (i.e. if we increase the number of instances in the training set, some categories can grow to larger size with the same δ value, since now the training points are closer to each other).

3.5. Discussion of the other ART parameters (α, $\bar{\rho}_a$, $\Delta\rho$)

We have already talked about the choice parameter α, when we elaborated on the μARTMAP parameter e. There we said that in our experiments we chose two values for the choice parameter, an α value of 0.01 and an α value of 0.001.

The baseline vigilance threshold $\bar{\rho}_a$ can be initialized within [0, 1]. Only in the first epoch does ρ_a explicitly depends on $\bar{\rho}_a$. From the second epoch, ρ_a is determined by the size of the most entropic node and $\Delta\rho$, according to (10). However, $\bar{\rho}_a$ is still important for μARTMAP since $\rho_a \geq \bar{\rho}_a$ in all epochs. If we set $\bar{\rho}_a = 0$ μARTMAP allows an arbitrarily large hyper-box to be created in the first epoch of training. On the other hand, if we set $\bar{\rho}_a = 1$ μARTMAP allows only zero-sized hyper-boxes, which means that every category in μARTMAP will represent a single pattern. In our experiments, we varied $\bar{\rho}_a$ within the set $\{0, 0.2, 0.4, 0.6, 0.8\}$.

$\Delta\rho$ is introduced only to make sure that the most entropic category cannot be created again after it is removed. Apparently, $\Delta\rho$ should not be set too large to avoid increasing ρ_a too quickly. If $\Delta\rho$ is too small, however, a category may be created with entropy close to that of the removed category for many consecutive epochs, which means that the total entropy may be reduced in a slow manner resulting in increased training times for μARTMAP. $\Delta\rho$ is also difficult to estimate since the size of the most entropic category in a certain epoch is obviously dependent on the distribution of the patterns. Fortunately, our experiments show that the performance of μARTMAP is not sensitive to the value of $\Delta\rho$. Thus we fix $\Delta\rho$ as 0.02 (also the choice in the original μARTMAP paper).

4. Experiments

We have performed a number of experiments with Safe μARTMAP. The purpose of these experiments was two-fold: First to compare Safe μARTMAP performance with the performance of other ART classifiers in the literature, including ART classifiers that claimed that they have addressed the category proliferation problem in Fuzzy ARTMAP. Secondly, to identify "optimal" settings of the network parameters in Safe μARTMAP, independently of the dataset, so that we can avoid the extensive experimentation required to identify a good set of parameters when Safe μARTMAP is confronted with a classification problem.

For the first part of our experiments, we have compared Safe μARTMAP with Fuzzy ARTMAP (FAM) (Carpenter et al., 1992) and Ellipsoidal ARTMAP (EAM) (Anagnostopoulos & Georgiopoulos, 2001; Anagnostopoulos, 2001). EAM is similar with FAM, except of the fact that EAM covers the input space with ellipsoids instead of hyper-boxes. Furthermore, we have compared Safe μARTMAP with the semi-supervised versions of FAM and EAM denoted ssFAM and ssEAM, respectively. The idea of semi-supervision is discussed in Anagnostopoulos et al. (2003, 2002), and in Verzi et al. (2001) (without referring to it as semi-supervision). Semi-supervision is a term referring to the fact that in the respective ART architectures categories are allowed to encode patterns that belong to different labels, provided that a certain threshold of mixture of labels is not exceeded. Finally, we have compared Safe μARTMAP with Gaussian ARTMAP (GAM), distributed Gaussian ARTMAP (dGAM), and their semi-supervised versions, called ssGAM and ssdGAM. Note that distributed Gaussian ARTMAP corresponds to the ART network that is referred to as Gaussian ARTMAP, by its originator Williamson (1996); the word distributed refers to the fact in the performance phase more than one node in the category representation layer of the Gaussian ARTMAP is activated to predict the label of the presented input pattern. We kept the name Gaussian ARTMAP for a Gaussian ARTMAP network which activates only the winning node in its performance phase.

In the sequel, we report results from both of these sets of experiments.

4.1. Databases

We experimented with both artificial and real databases. The specifics of these databases are given in Table 1.

Table 1
Databases used in the ARTMAP experiments

	Database name	# Training instances	# Validation instances	# Test instances	# Numerical attributes	# Classes (N_b)	Major class proportion (A_0)	Expected accuracy ($\hat{A}$)
1	G2c-05	500	5000	5000	2	2	1/2	0.95
2	G2c-15	500	5000	5000	2	2	1/2	0.85
3	G2c-25	500	5000	5000	2	2	1/2	0.75
4	G2c-40	500	5000	5000	2	2	1/2	0.6
5	G4c-05	500	5000	5000	2	4	1/4	0.95
6	G4c-15	500	5000	5000	2	4	1/4	0.85
7	G4c-25	500	5000	5000	2	4	1/4	0.75
8	G4c-40	500	5000	5000	2	4	1/4	0.6
9	G6c-05	504	5004	5004	2	6	1/6	0.95
10	G6c-15	504	5004	5004	2	6	1/6	0.85
11	G6c-25	504	5004	5004	2	6	1/6	0.75
12	G6c-40	504	5004	5004	2	6	1/6	0.6
13	MOD-IRIS	500	4800	4800	2	2	1/2	0.95
14	ABALONE	501	1838	1838	7	3	1/3	0.6
15	PAGE	500	2486	2487	10	5	0.832	0.95

(a) Gaussian Databases (G#c-##)

These are artificial databases, where we created 2-dimensional data, Gaussianly distributed, belonging to 2-class, 4-class, and 6-class problems. In each one of these databases we varied the amount of overlap of data belonging to different classes. In particular, we considered 5%, 15%, 25%, and 40% overlap. Note that 5% overlap means the optimal Bayesian Classifier would have 5% misclassification rate on the Gaussianly distributed data. There are a total of $3 \times 4 = 12$ Gaussian databases. We name the databases as "G#c-##" where the first number is the number of classes and the second number is the class overlap. For example, G2c-05 means the Gaussian database is a 2-class and 5% overlap database.

(b) Modified Iris Database (MOD-IRIS)

In this database we started from the IRIS dataset (Hettich, Blake, & Merz, 1998) of the 150-point 3-class problem. We eliminated the data corresponding to the class that is linearly separable from the others. Thus we ended up with 100 data-points. From the four input attributes of this IRIS dataset we focused on only two attributes (attribute 3 and 4) because they seem to have enough discriminatory power to separate the 2-class data. Finally, in order to create a reasonable size of dataset from these 100 points (so we can reliably perform cross-validation to identify the optimal Safe μARTMAP parameters) we created noisy data around each one of these 100 data-points (the noise was Gaussian of zero mean and small variance) to end up with additional 10,000 points. We named this database Modified Iris.

(c) Modified Abalone Database (ABALONE)

This database is originally used for prediction of the age of an abalone (Hettich et al., 1998). It contains 4177 instances, each with seven numerical attributes, one categorical attribute, and one numerical target output (age). We discarded the categorical attribute in our experiments, and grouped the target output values into three classes: eight and lower (class 1), 9–10 (class 2), 11 and greater (class 3). This grouping of output values has been reported in the literature before.

(d) Page Blocks Database (PAGE)

This database represents the problem of classifying the blocks of the page layout in a document (Hettich et al., 1998). One of the noteworthy points about this database is that its major class has a high probability of occurring (above 80%).

The data in each one of the above databases were split into a training set, a validation set and a test set. The percentage of classes in each one of these subsets resembled the percentage of classes in the original dataset. The summarized specifics of each one of these databases are depicted in Table 1. The training set was used to train the ART networks, the validation test was used to assess the performance of the ART networks for various settings of their parameter values, and the test set was used to report the performance of the "best performing" networks.

4.2. Parameter settings

For each database, we simulated Safe μARTMAP with the following settings for parameter $H_{\max}$:

$$H_{\max} = \{H_1, H_2, H_3, H_4, H_5\}$$
$$H_1 = \frac{1}{2}(H_L + H_2)$$
$$H_2 = \min\{H_{E1}, H_{E2}\}$$
$$H_3 = \frac{1}{2}(H_{E1} + H_{E2})$$
$$H_4 = \max\{H_{E1}, H_{E2}\}$$
$$H_5 = \begin{cases} H_U, & H_U > H_4 \\ 2H_U - H_3, & H_U = H_4. \end{cases}$$

The selection of the candidate values of $H_{\max}$ is actually difficult, because given the expected accuracy, we can only find out the possible range of H, and the actual entropy can be any value in that range, depending on the distribution of the data. Therefore, we have decided to experiment with some typical values for $H_{\max}$: we know that H_L, the lower bound of H, always causes over-training, and thus we set H_1, as the

smallest candidate value for $H_{\max}$, to be equal to $(H_L + H_2)/2$. The typical estimates, H_{E1}, H_{E2}, do not have a deterministic relative order (which means that one can be greater or less than the other). So, we introduced three candidate values H_2, H_3, H_4 dependent on H_{E1}, H_{E2}, and shown above. Finally, knowing that H_4 can be equal to H_U, the upper bound of $H_{\max}$, we define the last candidate value for $H_{\max}$ as always greater than H_4 to avoid experimentation with the same parameter values. Note that H_U is defined as the analytical upper bound given that the training accuracy equals the expected one. If we allow the training accuracy to fall below the expected one, H can be larger than H_U. We know that even when the training accuracy is below 80%, the test accuracy could be greater than 80% (see Fig. 3).

The settings for the rest of the Safe μARTMAP parameters ($h_{\max}$, $\bar{\rho}_a$, $\Delta\rho$, α, δ, and e) are provided below. The reasoning for these settings was provided in Section 3.

$$h_{\max} = \left\{0, \frac{1}{4}H_{\max}, \frac{1}{2}H_{\max}, H_{\max}, 2H_{\max}, \infty\right\}$$

$$\bar{\rho}_a = \left\{0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}\right\}$$

$$\Delta\rho = 0.02$$

$$\alpha = \{0.001, 0.01\}$$

$$\delta = \left\{\frac{1}{25}(1-\bar{\rho}_a), \frac{1}{5}(1-\bar{\rho}_a), (1-\bar{\rho}_a)\right\}$$

$$e = 0.05$$

Max number of epochs = 100.

We experimented with all the combinations of the above parameters, which amounted to $5 \times 6 \times 5 \times 2 \times 3 = 900$ combinations.

For the other ARTMAP architectures (see Chalasani (2005)), similar grid tests were performed, where the parameters were chosen from the combinations of the following values:

$$\bar{\rho}_a = \{0, 0.1, 0.2, \ldots, 0.9\}$$

$$\alpha = \{0.001, 0.01\}$$

$$\mu = \{0, 0.1, 0.2, \ldots, 1\}$$

$$D = \sqrt{M_a}/\mu \quad \text{(for EAM and ssEAM)}$$

$$\varepsilon = \{0, 0.1, 0.2, \ldots, 1\}$$

(for semi-supervised ARTMAPs)

$$\gamma = \{0.1, 0.2, \ldots, 1\}$$

(for GAM ssGAM, dGAM, and ssdGAM)

Max number of epochs = 100.

4.3. Experimental procedure — experimental results

As we have emphasized above, our experiments were divided into two parts.

In the *first* part, we compared Safe μARTMAP with other ARTMAP classifiers — Fuzzy ARTMAP (Carpenter et al., 1992), Ellipsoidal ARTMAP (Anagnostopoulos & Georgiopoulos, 2001; Anagnostopoulos, 2001), Gaussian ARTMAP and distributed Gaussian ARTMAP (Williamson, 1996), and their semi-supervised versions (Anagnostopoulos et al., 2003, 2002). The results of the other ARTMAP classifiers are obtained from Chalasani (2005).

For each database, we evaluated all the possible parameter combinations of Safe μARTMAP for a 100 different orders of the pattern list presentation (the performance of Safe μARTMAP depends on the order according to which patterns are presented in the training set). The 100 orders were fixed in all experiments and are exactly the same as those used to test the other ART algorithms (Fuzzy ARTMAP, Ellipsoidal ARTMAP, etc.) Therefore, for each database, we trained $900 \times 100 = 90{,}000$ Safe μARTMAP networks; we picked the network maximizing the following score:

$$\text{score} = \frac{A - A_0}{\hat{A} - A_0} 0.9^{(N_a/5N_b)^2} \tag{15}$$

where A is the accuracy of the trained ART network on the validation set, N_a is the number of categories formed in the training phase of Safe μARTMAP, and A_0, $\hat{A}$, and N_b are given in Table 1.

As can be seen, the score function is chosen so that its value does not change significantly for small values of the network size, N_a (when N_a is small, $\partial\text{score}/\partial N_a \approx 0$, which means that when the size of the network is small enough, its effect on the score is not significant). Furthermore, the score is chosen such that as N_a increases its value decreases (thus penalizing networks of large size). On the other hand, the relationship between score and network accuracy is linear.

The comparisons of Safe μARTMAP and the rest of the ART networks are depicted in Table 2. In Table 2, the first column is the index of the database that we are experimenting with. The second column is the actual databases name, as reported in earlier sections. Columns 3–11 of Table 2 contain the performance of the designated ART networks. The performance reported includes the accuracy on the test set and the number of categories created of the designated ART network that attained the highest value of score (this value was computed based on the accuracy of the network on the cross-validation set (the A value), and on the size of the network created (the N_a value)). According to Table 2 we can make some general statements. For one, Safe μARTMAP tends to yield a small network with high accuracy. Second, the size Safe μARTMAP compares very favorably (is smaller in most instances) than the size of every other ART network that we compared it to, especially with respect to the FAM, EAM, and dGAM. Third, the accuracy of Safe μARTMAP compares favorably with the accuracy of all the other ART networks that it is compared to; only some of the Gaussian networks seem to have an advantage over Safe μARTMAP for the database problems where the data are Gaussianly distributed, and for these type of problems Gaussian networks seem to be having an advantage over all the other ART networks. Finally, the two other ART networks that seem to be producing as good results as Safe μARTMAP are ssEAM and ssdGAM.

In the *second* part of our experiments, we elaborated on the search of the optimal parameter settings for Safe μARTMAP. We examined the parameters of the best networks using only

Table 2
Best performance of all ART algorithms

	Database name	Safe μAM		FAM		ssFAM		EAM		ssEAM		GAM		ssGAM		dGAM		ssdGAM	
1	G2c-05	95.22	2	90.80	14	94.90	2	91.72	26	94.94	2	94.06	4	94.48	4	95.22	4	95.22	2
2	G2c-15	85.00	2	77.68	47	84.80	3	77.88	79	85.20	2	84.86	6	85.04	2	84.76	8	85.02	2
3	G2c-25	74.98	2	64.36	75	74.60	2	65.06	123	74.50	2	74.88	6	75.10	2	74.90	7	75.10	2
4	G2c-40	61.40	3	53.84	110	61.34	3	53.58	177	60.98	2	59.64	12	61.30	3	60.30	9	61.32	3
5	G4c-05	95.04	4	92.84	21	94.10	7	92.96	24	94.14	4	94.84	10	94.80	4	94.84	10	94.80	4
6	G4c-15	83.28	4	77.52	55	81.40	11	78.12	76	83.20	4	84.00	18	84.24	9	84.00	18	84.20	9
7	G4c-25	74.50	4	67.06	101	70.80	9	66.58	110	72.72	4	73.74	49	72.32	21	74.60	46	74.96	35
8	G4c-40	59.76	5	48.52	127	58.48	14	49.58	161	55.62	13	58.08	36	59.10	14	58.92	36	59.40	14
9	G6c-05	93.57	9	91.85	26	91.42	11	92.30	23	93.80	7	94.49	12	94.40	8	94.68	13	94.84	6
10	G6c-15	80.92	6	76.23	58	81.11	7	76.09	85	81.80	6	84.67	19	84.35	13	85.03	19	83.87	11
11	G6c-25	70.74	13	66.66	87	69.62	15	63.74	124	71.10	7	73.24	30	72.86	20	73.65	32	73.22	20
12	G6c-40	58.03	11	51.40	196	56.35	17	50.69	193	54.21	17	58.51	70	55.65	13	59.03	70	55.50	13
13	MOD-IRIS	94.92	2	91.93	23	93.41	8	93.37	28	94.54	2	94.50	4	94.54	2	94.52	4	94.54	2
14	ABALONE	57.18	4	46.40	29	59.52	6	46.24	86	56.80	7	45.87	12	55.10	3	46.13	12	55.10	3
15	PAGE	88.82	6	83.27	10	90.63	3	76.71	34	89.54	3	85.52	9	89.34	5	85.52	9	89.34	5

μAM: Safe μARTMAP; FAM: Fuzzy ARTMAP; EAM: Ellipsoidal ARTMAP; GAM: Gaussian ARTMAP; dGAM: Distributed Gaussian ARTMAP; ss*: semi-supervised version.

the Gaussian databases (for which we know the exact value of $\hat{A}$). For each parameter combination and each Gaussian database, we set the score of the parameter combination as the maximum score of the 100 networks trained with that parameter combination (these 100 networks correspond to the 100 different orders of the training patterns during Safe μARTMAP's training). We use the maximum score instead of the average score because the orders also affect the network performance; there is no parameter setting that is good for all orders. Then, for every parameter combination we have 12 of these maximum scores (one maximum score for each of the 12 Gaussian datasets). We sum up these 12 maximum score numbers for every parameter combination, and then we rank these sums from highest to lowest. The five highest of these sums of maximum scores point us to the five default Safe μARTMAP parameter settings that can be used to experiment with any database of interest. To verify our claim, that the thus chosen 5 sets of default parameter values are good sets of parameters to experiment with, we are showing (in Table 3) Safe μARTMAP's performance (number of categories and accuracy on the test set) for the best Safe μARTMAP parameter values (set of columns designated as "Best in CV" in the table; note that "best in cross-validation" might not be "best on test set" although they are very close to each other) and for the five default parameter values that were identified from our experimentation of the Gaussian datasets, and explained above. The results reported in Table 3, and referred to as "Best in CV" correspond to the μARTMAP parameter settings and order of pattern presentation that attained the highest value of score for this database only. The results reported in Table 3 for the default parameter settings correspond to the order of pattern that attained the highest score. Each performance cell in Table 3 is made up of three numbers: the accuracy on the test set in percentage, the number of categories, and the number of epochs spent on training. An obvious observation from Table 3 is that the default parameters produce good results, compared to the best parameter setting. Second, the number of categories corresponding to the 5 default parameter settings, although not coinciding with the Best number of categories, have values that are close to the latter.

It is important to state, once more, that the identification of good, default parameter values for Safe μARTMAP is experimenting with $H_{\max}, h_{\max}, \bar{\rho}_a, \alpha, \delta$, and the order of training pattern presentation and this someone were to choose 5, 6, 5, 2, 3, and 100 distinct values for these parameters one would end up with the formidable task of training Safe μARTMAP 90,000 times for each problem (as we did in this paper). Instead by using default values for $H_{\max}, h_{\max}, \bar{\rho}_a, \alpha$ we would require to train Safe μARTMAP 3 (for δ) times 100 (for order of pattern presentations) times, or equivalently 300 times, which is a significant reduction in computational complexity to come up with a good performing Safe μARTMAP network.

Relying on the results reported in Table 3, we identified the following values as good default values to experiment with Safe μARTMAP for any classification problem. It is worth noting that these values are valid only if one allows Safe μARTMAP to train for reasonably large number of epochs (in our experiments we allowed Safe μARTMAP to train for up to 100 number of epochs).

$$H_{\max} = H_4, \quad h_{\max} = \infty, \quad \bar{\rho}_a = 0, \quad \alpha = 0.001. \tag{16}$$

We do not report a good default value for δ because its value depends on the size of the training set (and the relationship is not clear yet). Nevertheless the value of δ is constrained in the interval $[0, 1 - \bar{\rho}_a]$. Furthermore, from experimental observations we can state that as the size of the training set increases good default values for δ decrease. We are confident for the value of $H_{\max}$ since all the five default parameter combinations have this value (in fact, the best 65 combinations have this value). This result is not surprising, since H_4 is a good estimate of the entropy without over-training. The parameter settings $\bar{\rho}_a = 0$ and $\alpha = 0.001$ imply that we should allow a category to be very large in the first epoch, which is one

Table 3
Best parameter combinations

Rank	Best in CV			1			2			3			4			5		
$H_{\max}$	–			H_4			H_4			H_4			H_4			H_4		
$h_{\max}/H_{\max}$	–			∞			∞			1			∞			∞		
$\bar{\rho}_a$	–			0.4			0			0.2			0			0.2		
α	–			0.001			0.01			0.001			0.001			0.001		
$\delta/(1-\bar{\rho}_a)$	–			0.2			0.2			0.2			1			1		
G2c-05	95.22	2	1	95.16	2	14	95.14	2	4	95.20	2	1	95.20	3	10	95.20	3	10
G2c-15	85.00	2	1	85.06	2	4	84.98	3	15	85.06	2	1	85.24	2	27	85.24	2	27
G2c-25	74.98	2	1	74.96	2	16	74.96	3	18	74.18	2	1	75.02	3	8	75.02	3	8
G2c-40	61.40	3	1	61.54	4	8	61.34	4	18	61.44	3	10	61.32	4	32	61.32	4	32
G4c-05	95.04	4	22	94.82	4	25	94.36	6	50	94.64	4	1	94.46	6	48	94.46	6	48
G4c-15	83.28	4	20	81.74	6	44	84.18	7	65	83.58	9	82	83.64	9	61	83.64	9	61
G4c-25	74.50	4	44	74.78	5	37	75.06	6	52	75.06	4	48	75.02	6	49	75.02	6	49
G4c-40	59.76	5	39	59.26	4	52	59.76	5	39	58.84	5	41	59.72	7	37	59.72	7	37
G6c-05	93.57	9	9	93.09	10	85	91.87	9	74	93.23	10	58	93.53	13	93	93.53	13	93
G6c-15	80.92	6	1	81.18	12	100	81.87	13	100	81.16	14	76	82.27	12	100	82.27	12	100
G6c-25	70.74	13	88	71.18	13	83	69.54	14	85	69.76	11	100	69.16	13	90	69.16	13	90
G6c-40	58.03	11	100	56.77	16	100	56.45	13	81	56.41	13	100	56.30	14	77	56.30	14	77
MOD-IRIS	94.92	2	2	94.92	4	10	95.15	4	19	94.92	4	16	94.63	3	10	94.63	3	12
ABALONE	57.18	4	4	55.06	2	2	54.08	2	4	54.52	3	2	53.59	2	6	53.59	2	6
PAGE	88.82	6	17	88.34	5	10	92.32	5	24	89.14	8	35	89.75	4	11	89.75	4	11

of the beneficial features of Safe μARTMAP as mentioned in Section 2.

Although the optimal value of $h_{\max}$ seems unexpected, it can be explained as follows. This value allows a category to be very impure and tends to result in many more epochs of training because many impure categories must be removed in the future. In the first epoch, large categories will be created due to the small $\bar{\rho}_a$ value. In only a few epochs, the size of the categories will be almost entirely controlled by ρ_a. The number of categories will be very small at the beginning of training and it will grow slowly afterwards, until the total entropy is no smaller than $H_{\max}$. Therefore, through this training process, the minimum number of categories may be achieved. Of course, sufficient epochs of training must be allowed, or otherwise the training process would be terminated prematurely and the network performance would be even worse than when $h_{\max} = 0$. In contrast, setting $h_{\max} = 0$ will cause a large number of categories to be created in the first epoch, including many trivial categories. In this case, the training process may be completed in only one epoch, resulting in a Safe μARTMAP network that may still be over-trained, exhibiting poor generalization.

5. Conclusions

Safe μARTMAP is one of the recently introduced ARTMAP architectures, which can produce small size classifiers with high accuracy. Safe μARTMAP effectively addresses the category proliferation problem in Fuzzy ARTMAP. One of the important issues in using Safe μARTMAP is the correct selection of its many parameters.

In this paper, we have studied the effect of these Safe μARTMAP parameters, both theoretically and experimentally. Furthermore, we have identified a procedure that allowed us to define *good* default Safe μARTMAP parameter values, independently of the database used, despite the obvious fact that the *best* Safe μARTMAP parameter values are database dependent. Even in case our default values are not good enough, they can serve as a starting point for further or finer search. In any case, this is a significant simplification for anyone experimenting with Safe μARTMAP on new datasets, especially when the datasets are large and training a single Safe μARTMAP network is costly. Furthermore, it is also very beneficial in cases when the dataset is small and expensive cross-validation procedures (e.g. one-hold-out cross-validation) are used to identify good network parameter settings.

In this paper, we have also compared the performance of Safe μARTMAP with a number of other ART classifiers, including the ones that have been reported in the literature and claim that they also address the category proliferation problem in Fuzzy ARTMAP. The result from this experimentation is that Safe μARTMAP outperforms Fuzzy ARTMAP (FAM), Ellipsoidal ARTMAP (EAM), and Gaussian ARTMAP (GAM), and it exhibits comparable performance with semi-supervised EAM and distributed GAM.

Finally, it is worth pointing out that our performance comparison of various ART algorithms and the identification of good, default parameter values for Safe μARTMAP relied on a performance measure (score) that takes into consideration both the accuracy of the network on a cross-validation set and the size of the network that training creates. Despite its obvious benefits this is an approach that has not been quantified in the ART literature.

Acknowledgments

This work was supported in part by a National Science Foundation (NSF) grant CRCD: 0203446, and the National Foundation grant DUE 05254209. Georgios C. Anagnostopoulos and Michael Georgiopoulos acknowledge the partial support from

the NSF grant CCLI 0341601. We would also like to acknowledge the help from Dr. Eduardo Gomez-Sanchez who provided us information about some of the subtle details pertaining to Safe μARTMAP.

Appendix A. Estimates of $H_{\max}$

A.1. Preliminaries

Assume the expected accuracy $\hat{A}$ is given and $1/N_b < \hat{A} \leq 1$. Before we study the relationship between $H_{\max}$ and $\hat{A}$, we have to define four accuracies: the accuracy on the training set produced by using the entries of the $\mathbf{W}^{ab}$ matrix (designated by A_W^{Train}), the accuracy on the training set produced by using the entries of the $\mathbf{V}^{ab}$ matrix (designated by A_V^{Train}), the accuracy on the validation set produced by using the entries of the $\mathbf{W}^{ab}$ matrix (designated by A_W^{Val}), and the accuracy on the test set produced by using the entries of the $\mathbf{W}^{ab}$ matrix (designated by A_W^{Test}). After training, only the $\mathbf{W}^{ab}$ matrix is used to produce the classification results. For this reason, we do not examine the accuracy on the validation/test set using the entries of the $\mathbf{V}^{ab}$ matrix; we examine A_V^{Train} because the total entropy H is computed based on the entries of the $\mathbf{V}^{ab}$ matrix instead of the entries of the $\mathbf{W}^{ab}$ matrix.

It is a well-known result that when the accuracy on the training set is increased too much, the accuracy on the test set will drop since the network is over-trained. Here we do not consider the case where the database is so small that the training set might not be representative. In Figs. 3 and 4 we show the relationships between these accuracies for the 2-class Gaussian dataset problems (similar plots are valid for the other Gaussian problems). In particular, Fig. 3 shows the relationship between A_V^{Train} and A_W^{Test}. Furthermore, Fig. 4 shows the relationship between A_W^{Val} and A_W^{Test}. Some of the observations that can be extracted from these two figures are provided below.

(1) When $A_V^{\text{Train}} < \hat{A}$, $\max A_W^{\text{Test}}$ (the A_W^{Test} value of the network with the best parameter settings) increases with A_V^{Train} and $A_V^{\text{Train}} < \max A_W^{\text{Test}} < \hat{A}$; when $A_V^{\text{Train}} = \hat{A}$, $\max A_W^{\text{Test}} \approx \hat{A}$; when $A_V^{\text{Train}} > \hat{A}$, $\max A_W^{\text{Test}}$ decreases with A_V^{Train} and $\max A_W^{\text{Test}} < \hat{A}$.
(2) $A_W^{\text{Test}} \approx A_W^{\text{Val}}$. This is reasonable since both the test set and the validation set are unseen by the network, and they represent the same problem.

It is clear that the training algorithm should be terminated when A_V^{Train} reaches $\hat{A}$.

A.2. Theoretical upper bound

Next, we try to estimate H given that $A_V^{\text{Train}} = \hat{A}$. In the following part, we find the maximum and the minimum of H. Let $p_j = \frac{|\mathbf{V}_j^{ab}|}{|\mathbf{V}^{ab}|}$ and $p_{jk} = \frac{\mathbf{V}_{jk}^{ab}}{|\mathbf{V}_j^{ab}|}$. The accuracy of node j on the training set produced by using the entries of matrix $\mathbf{V}^{ab}$ can be

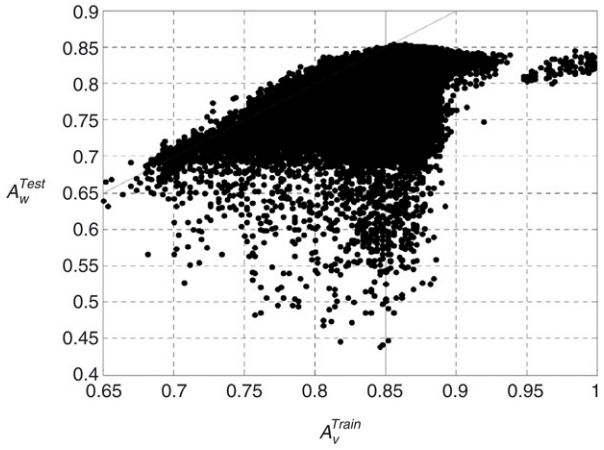

Fig. 3. Accuracy on test set versus accuracy on training set.

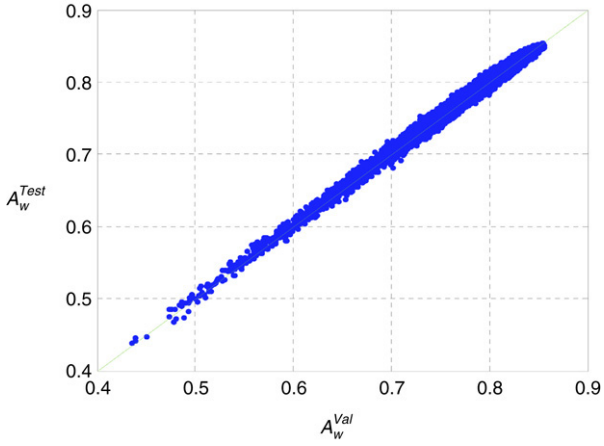

Fig. 4. Accuracy on test set versus accuracy on validation set.

expressed as $A_j = \max_k p_{jk}$. The maximum problem can be described as:

$$\text{Maximize} \quad H = -\sum_{j=1}^{N_a}\left(p_j \sum_{k=1}^{N_b} p_{jk}\log_2 p_{jk}\right).$$

$$\text{Subject to} \quad \sum_{j=1}^{N_a} p_j = 1$$

$$\sum_{k=1}^{N_b} p_{jk} = 1 \quad \text{for } j = 1, 2, \ldots N_a$$

$$\sum_{j=1}^{N_a}(p_j \max_k p_{jk}) = \hat{A}$$

$$0 \leq p_j \leq 1 \quad \text{for } j = 1, 2, \ldots N_a$$

$$0 \leq p_{jk} \leq 1 \quad \text{for all } j \text{ and all } k.$$

To simplify the problem, we can first maximize the entropy of each node given A_j by adjusting p_{jk}, and then maximize H by adjusting p_j and A_j (and N_a, if necessary). Without loss of generality, we can assume $p_{j1} = \max_k p_{jk} = A_j$ for all j. Since the function $f(x) = -x\log_2 x$ is strictly concave in the interval $(0, +\infty)$, $\frac{\sum_{k=2}^{N_b} f(p_{jk})}{N_b - 1} \leq f(\frac{\sum_{k=2}^{N_b} p_{jk}}{N_b - 1})$, i.e., $-\sum_{k=1}^{N_b} p_{jk}\log_2 p_{jk} \leq -(1-A_j)\log_2 \frac{1-A_j}{N_b-1}$. The equality

holds if and only if $p_{j2} = p_{j3} = \cdots = p_{jk} = \frac{1-A_j}{N_b-1}$. In this case, we can simplify H as:

$$\begin{aligned} H &= -\sum_{j=1}^{N_a} p_j \left[A_j \log_2 A_j + (1-A_j) \log_2 \frac{1-A_j}{N_b-1} \right] \\ &= -\sum_{j=1}^{N_a} p_j [A_j \log_2 A_j + (1-A_j) \log_2 (1-A_j)] \\ &\quad + \sum_{j=1}^{N_a} p_j (1-A_j) \log_2 (N_b-1) \\ &= -\sum_{j=1}^{N_a} p_j [A_j \log_2 A_j + (1-A_j) \log_2 (1-A_j)] \\ &\quad + (1-\hat{A}) \log_2 (N_b-1) \end{aligned}$$

Therefore, the problem reduces to:

$$\begin{aligned} \text{Maximize} \quad & H = -\sum_{j=1}^{N_a} p_j [A_j \log_2 A_j + (1-A_j) \\ & \qquad \times \log_2 (1-A_j)] + (1-\hat{A}) \log_2 (N_b-1). \\ \text{Subject to} \quad & \sum_{j=1}^{N_a} p_j = 1 \\ & \sum_{j=1}^{N_a} p_j A_j = \hat{A} \\ & 0 \le p_j \le 1 \quad \text{for } j = 1, 2, \ldots N_a \\ & \frac{1}{N_b} \le A_j \le 1 \quad \text{for } j = 1, 2, \ldots N_a. \end{aligned}$$

Using again the concavity of $f(x) = -x \log_2 x$, we have:

$$\sum_{j=1}^{N_a} p_j f(A_j) \le f\left(\sum_{j=1}^{N_a} p_j A_j \right),$$

$$\sum_{j=1}^{N_a} p_j f(1-A_j) \le f\left(\sum_{j=1}^{N_a} p_j (1-A_j) \right)$$

which means

$$\begin{aligned} H \le & -\hat{A} \log_2 \hat{A} - (1-\hat{A}) \log_2 (1-\hat{A}) + (1-\hat{A}) \\ & \times \log_2 (N_b-1). \end{aligned}$$

The equality holds if and only if $A_j = \hat{A}$ for all j. Thus, we get the following theoretical upper bound for H:

$$H_U = -\hat{A} \log_2 \hat{A} - (1-\hat{A}) \log_2 \frac{1-\hat{A}}{N_b-1}.$$

A.3. Theoretical lower bound

Following the same approach used to derive the theoretical upper bound for H, we can extract the theoretical lower bound for H. We first minimize $-\sum_{k=1}^{N_b} p_{jk} \log_2 p_{jk}$ subject to $A_j = p_{j1} \ge p_{j2} \ge p_{j3} = \cdots = p_{jk} \ge 0$ and $\sum_{k=1}^{N_b} p_{jk} = 1$. Based on the concavity of the function $f(x) = -x \log_2 x$, we know that $-\sum_{k=1}^{N_b} p_{jk} \log_2 p_{jk}$ is minimized when $p_{j1} = p_{j2} \cdots = p_{jn} = A_j$ and $p_{jn+1} = 1 - nA_j$, where $n = \lfloor 1/A_j \rfloor$. If two p_{jk1} and p_{jk2} are less than A_j, we can construct an example with $p'_{jk1} = \min(A_j, p_{jk1} + p_{jk2})$ and $p'_{jk2} = p_{jk1} + p_{jk2} - p'_{jk1}$ which leads to lower entropy. Therefore, the minimum of $-\sum_{k=1}^{N_b} p_{jk} \log_2 p_{jk}$ is $-nA_j \log_2 A_j - (1-nA_j) \log_2 (1-nA_j)$. The floor function in the expression of n, however, makes our analysis somehow difficult since it is not continuous. Note that $-nA_j \log_2 A_j - (1-nA_j) \log_2 (1-nA_j) \ge -\log_2 A_j$ (the equality holds if and only if $1/A_j$ is an integer). We use $-\log_2 A_j$ as the lower bound for convenience. The problem becomes:

$$\begin{aligned} \text{Minimize} \quad & H = -\sum_{j=1}^{N_a} p_j \log_2 A_j. \\ \text{Subject to} \quad & \sum_{j=1}^{N_a} p_j = 1 \\ & \sum_{j=1}^{N_a} p_j A_j = \hat{A} \\ & 0 \le p_j \le 1 \quad \text{for } j = 1, 2, \ldots N_a \\ & \frac{1}{N_b} \le A_j \le 1 \quad \text{for } j = 1, 2, \ldots N_a. \end{aligned}$$

The function $g(x) = -\log_2 x$ is strictly convex in the interval $(0, +\infty)$. Hence,

$$H = \sum_{j=1}^{N_a} p_j g(A_j) \ge g\left(\sum_{j=1}^{N_a} p_j A_j \right) = g(\hat{A}) = -\log_2 \hat{A},$$

where the equality holds if and only if $A_j = \hat{A}$ for all j. We, therefore, obtain the theoretical lower bound of H:

$$H_L = -\log_2 \hat{A}.$$

A.4. Typical case 1

In most cases, both the theoretical upper bound and lower bound are far from the actual value of H, since they require that many constraints must be met, as shown above. Here we just consider two typical cases to estimate H.

In the first case, the accuracies of all the categories are either 1 or $\frac{1}{N_b}$. Thus,

$$H = -\sum_{j=1}^{K} p_j \log_2 N_b$$

$$\sum_{j=1}^{N_a} p_j = 1$$

$$\sum_{j=1}^{K} p_j \frac{1}{N_b} + \sum_{j=K+1}^{N_b} p_j = \hat{A}.$$

It is not difficult to solve the above equations and find the corresponding H value. We denote the resulting H value by H_{E1} and we are providing it below.

$$H_{E1} = \frac{N_b(1-\hat{A})}{N_b-1} \log_2 N_b.$$

A.5. Typical case 2

In the second case, $A_j = \hat{A}$ for all j, and $p_{jk} = p^{k-1}\hat{A}$, where p is a constant in the interval $[0, 1)$ (it cannot be one because $\hat{A} > \frac{1}{N_b}$) and satisfies $\sum_{k=1}^{N_a} p_{jk} = 1$, i.e., $\frac{1-p}{1-p^{N_b}} = \hat{A}$. This means all the class fractions make a geometric progress. Solving for p is not difficult: when $2 \le N_b \le 5$, we can solve this equation analytically; when $N_b > 5$ and $\hat{A} \ge 0.5$, $p \approx 1 - \hat{A}$; when $N_b > 5$ and $\hat{A} < 0.5$, we can solve the equation numerically, which is not difficult since $\sum_{k=1}^{N_b} p^{k-1}$ is monotonically increasing in p.

In this case, the resulting H value, designated by H_{E2}, can be computed as follows:

$$\begin{aligned}
H_{E2} &= -\sum_{k=1}^{N_b} p^{k-1}\hat{A}\log_2(p^{k-1}\hat{A}) \\
&= -\sum_{i=0}^{N_b-1} p^i\hat{A}\log_2(p^i\hat{A}) \\
&= -\sum_{i=0}^{N_b-1} p^i\hat{A}(i\log_2 p + \log_2\hat{A}) \\
&= -\hat{A}(\log_2 p)\sum_{i=0}^{N_b-1} ip^i - \hat{A}(\log_2\hat{A})\sum_{i=0}^{N_b-1} p^i \\
&= -\hat{A}(\log_2 p)\frac{p - N_b p^{N_b} + (N_b-1)p^{N_b+1}}{(1-p)^2} \\
&\quad - \hat{A}(\log_2\hat{A})\frac{1-p^{N_b}}{1-p}.
\end{aligned}$$

Since $\frac{1-p}{1-p^{N_b}} = \hat{A}$, $p^{N_b} = 1 - \frac{1-p}{\hat{A}}$, it is not difficult to simplify the above expression to obtain:

$$H_{E2} = -\frac{N_b(1-\hat{A}) - (N_b-1)p}{1-p}\log_2 p - \log_2\hat{A}.$$

A.6. Relationship among the estimates

Based on our aforementioned arguments we obtained four estimates for H, repeated below:

$$H_U = -\hat{A}\log_2\hat{A} - (1-\hat{A})\log_2\frac{1-\hat{A}}{N_b-1}$$

$$H_{E1} = \frac{N_b(1-\hat{A})}{N_b-1}\log_2 N_b$$

$$\begin{aligned}
H_{E2} &= -\hat{A}(\log_2 p)\sum_{i=0}^{N_b-1} ip^i - \log_2\hat{A} \\
&= -\frac{N_b(1-\hat{A}) - (N_b-1)p}{1-p}\log_2 p - \log_2\hat{A}
\end{aligned}$$

$$H_L = -\log_2\hat{A}.$$

It is not difficult to prove the following:

(1) H_U, H_{E1}, H_{E2}, H_L are all decreasing functions of $\hat{A}$. When $\hat{A} = \frac{1}{N_b}$ or $\hat{A} = 1$, $H_U = H_{E1} = H_{E2} = H_L$; when $N_b = 2$, $H_{E2} = H_U$. In all other cases, $H_U > H_{E1} > H_L$ and $H_U > H_{E2} > H_L$.

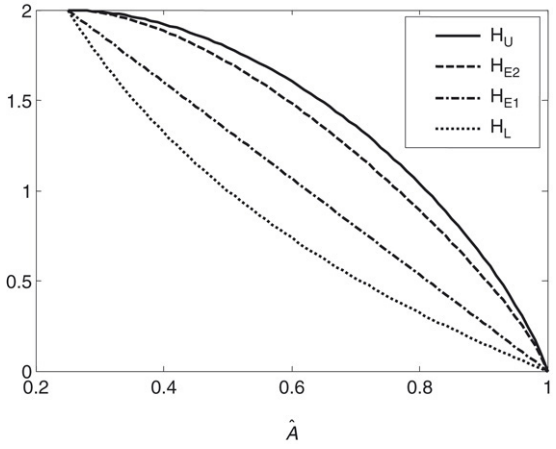

Fig. 5. Four typical H values with $N_b = 4$.

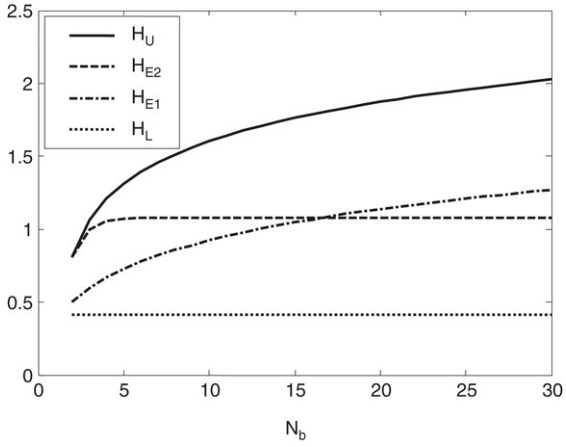

Fig. 6. Four typical H values with $\hat{A} = 0.75$.

(2) H_{E2} is an increasing function of N_b. When $N_b = 2$, $H_{E2} = H_0$, where $H_0 = -\hat{A}\log_2\hat{A} - (1-\hat{A})\log_2(1-\hat{A})$. $\lim_{N_b\to\infty} H_{E2} = \frac{1}{\hat{A}}H_0$.

(3) H_{E1} is an increasing function of N_b. When $N_b = 2$, $H_{E1} \le H_0$ (the equality holds iff $\hat{A} = 0.5$ or $\hat{A} = 1$); $\lim_{N_b\to\infty} H_{E1}$ is unbounded.

(4) H_U is an increasing function of N_b. When $N_b = 2$, $H_U = H_0$; when $N_b = 2^{H_0} + 1$, $H_U = \frac{1}{\hat{A}}H_0$; $\lim_{N_b\to\infty} H_U$ is unbounded.

The aforementioned H values (i.e., H_L, H_U, H_{E1}, H_{E2}) are plotted with respect to N_b and $\hat{A}$ in Figs. 5 and 6, as $\hat{A}$ varies and N_b is fixed (Fig. 5), and as N_b varies and $\hat{A}$ is kept fixed (Fig. 6).

References

Anagnostopoulos, G. C., Bharadwaj, M., Georgiopoulos, M., Verzi, S. J., & Heileman, G. L. (2003). Exemplar-based pattern recognition via semi-supervised learning. In *Proc. of the international joint conference on neural networks: Vol. 4* (pp. 2782–2787).

Anagnostopoulos, G. C., & Georgiopoulos, M. (2001). Ellipsoid ART and ARTMAP for incremental clustering and classification. In *Proc. of the IEEE-INNS international joint conference on neural networks* (pp. 1221–1226).

Anagnostopoulos, G. C., Georgiopoulos, M., Verzi, S. J., & Heileman, G. L. (2002). Reducing generalization error and category proliferation in ellipsoid ARTMAP via tunable misclassification error tolerance: Boosted Ellipsoid ARTMAP. In *Proc. of the international joint conference on neural networks: Vol. 3* (pp. 2650–2655).

Anagnostopoulos, G. C. (2001). Novel approaches in adaptive resonance theory for machine learning. Doctoral thesis. Orlando, Florida: University of Central Florida.

Carpenter, G. A., Grossberg, S., Markuzon, N., & Reynolds, J. H. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps. *IEEE Transactions on Neural Networks*, *3*(5), 698–713.

Carpenter, G. A., & Milenova, B. (1998). Distributed ARTMAP: A neural network for fast distributed supervised learning. *Neural Networks*, *11*(2), 323–336.

Chalasani, R. (2005). Optimization of network parameters and semi-supervision in Gaussian ART architectures. M.S. thesis. Orlando, Florida: University of Central Florida.

Charalampidis, D., Kasparis, T., & Georgiopoulos, M. (2001). Classification of noisy signals using Fuzzy ARTMAP neural networks. *IEEE Transactions on Neural Networks*, *12*(5), 1023–1036.

Gomez-Sanchez, E., Dimitriadis, Y. A., Cano-Izquierdo, J. M., & Lopez-Coronado, J. (2002). μARTMAP: Use of mutual information for category reduction in Fuzzy ARTMAP. *IEEE Transactions on Neural Networks*, *13*(1), 58–69.

Gomez-Sanchez, E., Dimitriadis, Y. A., Cano-Izquierdo, J. M., & Lopez-Coronado, J. (2001). Safe-μARTMAP: A new solution for reducing category proliferation in Fuzzy ARTMAP. In *Proc. of the IEEE international joint conference on neural networks: Vol. 2* (pp. 1197–1202).

Grossberg, S. (1976). Adaptive pattern recognition and universal recoding II: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, *23*, 187–202.

Hettich, S., Blake, C. L., & Merz, C. J. (1998). *UCI Repository of machine learning databases*. Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Kasuba, T. (1993). Simplified Fuzzy ARTMAP. *AI Expert*, 18–25.

Koufakou, A., Georgiopoulos, M., Anagnostopoulos, G. C., & Kasparis, T. (2001). Cross-validation in Fuzzy ARTMAP for large databases. *Neural Networks*, *14*, 1279–1291.

Marriott, S., & Harrison, R. F. (1995). A modified Fuzzy ARTMAP architecture for the approximation of noisy mappings. *Neural Networks*, *8*(4), 619–641.

Parrado-Hernandez, E., Gomez-Sanchez, E., & Dimitriadis, Y. A. (2003). Study of distributed learning as a solution to category proliferation in Fuzzy ARTMAP-based neural systems. *Neural Networks*, (16), 1039–1057.

Taghi, M., Bagmisheh, V., & Pavesic, N. (2003). A fast simplified Fuzzy ARTMAP network. *Neural Processing Letters*, *17*(3), 273–316.

Verzi, S. J., Georgiopoulos, M., Heileman, G. L., & Healy, M. (2001). Rademacher penalization applied to Fuzzy ARTMAP and Boosted ARTMAP. In *Proc. of the IEEE-INNS international joint conference on neural network* (pp. 1191–1196).

Williamson, J. R. (1997). A constructive, incremental-learning network for mixture modeling and classification. *Neural Computation*, *9*, 1517–1543.

Williamson, J. R. (1996). Gaussian ARTMAP: A neural network for fast incremental learning of noisy multi-dimensional maps. *Neural Networks*, *9*(5), 881–897.