Available online at www.sciencedirect.com
ScienceDirect
Neural Networks 20 (2007) 874–892
Neural Networks
www.elsevier.com/locate/neunet

GFAM: Evolving Fuzzy ARTMAP neural networks

Ahmad Al-Daraiseh[a], Assem Kaylani[a], Michael Georgiopoulos[a,*], Mansooreh Mollaghasemi[b], Annie S. Wu[a], Georgios Anagnostopoulos[c]

[a] *School of EECS, University of Central Florida, Orlando, FL 32816-2786, United States*
[b] *Department of IEMS, University of Central Florida, Orlando, FL 32816-2786, United States*
[c] *Department of ECE, Florida Institute of Technology, Melbourne, FL 32901-6975, United States*

Received 8 February 2006; accepted 23 May 2007

Abstract

This paper focuses on the evolution of Fuzzy ARTMAP neural network classifiers, using genetic algorithms, with the objective of improving generalization performance (classification accuracy of the ART network on unseen test data) and alleviating the ART category proliferation problem (the problem of creating more than necessary ART network categories to solve a classification problem). We refer to the resulting architecture as GFAM. We demonstrate through extensive experimentation that GFAM exhibits good generalization and is of small size (creates few ART categories), while consuming reasonable computational effort. In a number of classification problems, GFAM produces the optimal classifier. Furthermore, we compare the performance of GFAM with other competitive ARTMAP classifiers that have appeared in the literature and addressed the category proliferation problem in ART. We illustrate that GFAM produces improved results over these architectures, as well as other competitive classifiers.

Keywords: Machine learning; Classification; ARTMAP; Genetic algorithms; Genetic operators; Category proliferation

1. Introduction

The Adaptive Resonance Theory (ART) was developed by Grossberg (1976). One of the most celebrated ART architectures is Fuzzy ARTMAP (FAM) (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 FAM possesses is that it can solve arbitrarily complex classification problems, it converges quickly to a solution (within a few presentations of the list of 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 on-line fashion (new input/output patterns can be learned by the system without re-training with the old input/output patterns), and it produces answers that can be explained with relative ease.

One of the limitations of FAM that has been repeatedly reported in the literature is the category proliferation problem. This refers to the creation of a relatively large number of categories to represent the training data. Categories are the hidden nodes (or units) in a FAM neural network. Each node is mapped to a specific class. The creation of a large number of categories means poor compression of the training data. Quite often the category proliferation problem, observed in FAM, is connected with the issue of overtraining. Overtraining happens when FAM 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). Also, it has been related to several limitations of FAM, such as the representative inefficiency of the hyperbox categories or the excessive triggering of the match tracking mechanism due to existence of noise.

In this paper, we propose the use of genetic algorithms (Goldberg, 1989) to solve the category proliferation problem, while improving the generalization performance in FAM. We

* Corresponding author.
E-mail addresses: creepymaster@yahoo.com (A. Al-Daraiseh), akaylani@gmail.com (A. Kaylani), michaelg@mail.ucf.edu (M. Georgiopoulos), mollagha@mail.ucf.edu (M. Mollaghasemi), aswu@cs.ucf.edu (A.S. Wu), georgio@fit.edu (G. Anagnostopoulos).

0893-6080/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2007.05.006

refer to the resulting architecture as GFAM. In our work here, we use GAs to evolve simultaneously the weights, as well as the topology of the FAM neural networks. We start with a population of trained FAMs, whose number of nodes in the hidden layer and the values of the interconnection weights converging to these nodes are fully determined (at the beginning of the evolution) by the ART's training rules. To this initial population of FAM networks, GA operators are applied to modify these trained FAM architectures (i.e., number of nodes in the hidden layer, and values of the interconnection weights) in a way that encourages better generalization and smaller size architectures.

It is worth reminding the reader that, as with many neural network architectures, the knowledge in Fuzzy ARTMAP is stored in its interconnection weights that have a very interesting geometrical interpretation (see Anagnostopoulos and Georgiopoulos (2002), Carpenter, Grossberg, and Rosen (1991)). In particular, the interconnection weights in Fuzzy ARTMAP (converging to the nodes in the hidden layer) represent the lower and upper end-points of hyper-rectangles (referred to as categories) that enclose within their boundaries clusters of data that are mapped to the same label.

Eventually, the evolution of these trained FAMs produces a FAM architecture, referred to as GFAM, which has better generalization performance and smaller size than the FAMs that we started with in the initial FAM population. GFAM is the FAM network that at the last generation of the evolutionary process attained the highest fitness function value. In the evolution of FAM trained networks, in addition to the common GA operators, such as reproduction, crossover, and mutation, we used a unique (and needed) genetic operator, referred to as the Cat_{del} operator. This operator, by allowing us to destroy FAM categories, leads us to FAM networks of smaller size.

Our results show that GFAM performed well on a number of classification problems, and on a few of them it performed optimally. Furthermore, GFAM networks were found to be superior to a number of other ART networks (ssFAM, ssEAM, ssGAM, safe micro-ARTMAP) that have been introduced into the literature to address the category proliferation problem in ART. More specifically, GFAM gave a better generalization performance and a smaller than, or equal, size network (in almost all problems tested), compared to these other ART networks, requiring reduced computational effort to achieve these advantages. More specifically, in some instances the difference in classification performance of GFAM with these other ART networks quite significant (as high as 10%). Also, in some instances the ratio of the number of categories created by these other ART networks, compared to the categories created by GFAM was large (as high as 5).

The organization of this paper is as follows: In Section 2 we present a literature review relevant to some of the issues addressed in this paper. In Section 3 we emphasize some of the basics of the Fuzzy ARTMAP architecture. In Section 4 we describe all the necessary elements pertinent to the evolution of the Fuzzy ARTMAP architecture. In Section 5, we describe the experiments and the datasets used to assess the performance of GFAM, we assess the performance of GFAM, and we offer performance comparisons between GFAM and other ART architectures that were proposed as solutions for the category proliferation problem in FAM. Also, in Section 5 we compare GFAM with other non-ART based classifiers (although the comparison is not comprehensive). In Section 6, we summarize our contribution, findings, and we provide directions for future research.

2. Literature review

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, Georgiopoulos, Verzi, and Heileman (2002), Anagnostopoulos, Bharadwaj, Georgiopoulos, and Heileman (2003), and Gomez-Sanchez, Dimitriadis, Cano-Izquierdo, and Lopez-Coronado (2001, 2002) Verzi, Heileman, Georgiopoulos, 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), 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.

Genetic algorithms have been extensively used to evolve artificial neural networks. Genetic algorithms (GAs) are a class of population-based stochastic search algorithms that are developed from ideas and principles of natural evolution. An important feature of these algorithms is their population based search strategy. Individuals in a population compete and exchange information with each other in order to perform certain tasks. For a thorough exposition of the available research literature on evolving neural networks the interested reader is advised to consult Yao (1999). In Yao (1999), the author distinguishes three different strategies in evolving neural networks. The first strategy is the one used to search for the weights of the neural network (see for example Sexton, Dorsey, and Jonson (1998)), the second one is the one used to design the structure of the network (see for example Marin and Sandoval (1993), where only the structures are evolved, and Yao and Liu (1998), where both the structure and the interconnection weights are evolved), and the third one is the one where the learning rules of the neural network are evolved (see Hancock, Smith, and Phillips (1991)). Furthermore, GAs may also be used to select the features that are input to the neural network. Since the pioneering work by Siedlecki and Sklansky (Siedlecki & Sklansky, 1989), genetic algorithms have been used for

many selection problems using neural networks (Brotherton & Simpson, 1995; Yang & Honavar, 1998), and other classifiers, such as decision trees (Bala, De Jong, Huang, Vafaie, & Wechsler, 1996), k-nearest neighbors (Kelly Jr & Davis, 1991; Punch, Goodman, ChiaShun, Hovland, & Enbody, 1993), and naïve Bayes classifiers (Inza, Larranaga, Etxeberria, & Sierra, 1999; Cantu-Paz, 2002).

As has been verified in the literature, the topology of the neural network is crucial to its performance. If a network has too few nodes, it might not be able to learn the required task. On the other hand, if the network has too many nodes, it may over-fit the training data and thus exhibit poor generalization. Miller, Todd and Hedge (Miller, Todd, & Hedge, 1989) defined two major approaches to using GAs to design the topology of the neural networks. The first approach uses direct encoding to specify every connection of the network, or to evolve an indirect specification of the connectivity. The direct encoding GA approach implies that every connection between every node be directly represented in a chromosomal string. Direct encoding has been used effectively to prune neural networks with good results (Whitley, Starkweather, & Bogart, 1990; Hancock, 1992). On the other hand, a simple indirect encoding method is to commit to a particular topology (e.g., feed-forward or recurrent NN) and a learning algorithm (e.g. back-prop learning algorithm), and then use a GA to find the parameter values that complete the network specification. For example, the GA in the feed-forward neural network approach can search for the number of layers and the number of units (nodes) per layer. The indirect encoding scheme is far more sophisticated while being theoretically capable of representing complicated topologies with finesse. It encodes the most important parameters and leaves the remainder to be determined elsewhere. Harp, et al. (see Harp, Samad, and Guha (1989)) used segments of two parts in an encoding scheme entitled blueprints. The first segment held parameter specifications including address, organization and number of nodes, and learning parameters associated with the nodes. The second segment described the connections between themselves by specifying the density between the current area and the target area, the target's area address, organization of the connections, and parameters of learning associated with the connection weights.

To the best of our knowledge, work that utilizes GAs and ART neural network architectures is rather limited. For instance, in Liu, Liu, Liu, Zhang, and Wu (2003), a GA algorithm was employed to appropriately weigh attributes of input patterns before they were fed into the input layer of Fuzzy ARTMAP. The results reported reveal that this attribute weighting was beneficial because it produced a trained ART architecture of improved generalization. In our work, we do not use the GAs to optimize the weight with which its attribute of the input patterns is affecting the input layer of the ART architecture. Instead, as can be seen, we use GAs to optimize the topology and the weights of a trained ART architecture. Furthermore, in Burton and Vladimirova (1997), a Fuzzy ART neural network was employed as a GA fitness function evaluator, however the brevity of the published paper did not allow for the discussion of the details pertinent to this work.

In our work here, we use GAs to evolve simultaneously the weights as well as the topology of the Fuzzy ARTMAP neural networks. In contrast to the feed-forward neural networks that have been extensively evolved, the Fuzzy ARTMAP neural network has a number of topological constraints, such as (a) it consists of one hidden layer of nodes, called the category representation layer, and (b) every interconnection weight value from every node of the input layer to a node in the hidden layer is important (representing the minimum or the maximum of the values of input patterns across every dimension that were encoded by this node).

It is apparent that, in evolving neural network architectures, one has to decide on the genotype representation scheme for the neural network architecture under consideration, on the genetic operators used to evolve these neural network architectures, and on the fitness function used to guide this evolution. In this paper we address these issues in a manner that fits the characteristics of the FAM neural network and our ultimate objective of reducing category proliferation in FAM, while preserving good generalization performance. In addition to successfully addressing the issues related with the evolution of FAM structures, mentioned above, we also compare in this paper the final product of FAM's evolution with other approaches proposed in the ART literature that also addressed the category proliferation problem in ART. This comparison is based on the accuracy of the architectures and size of the architectures produced by these techniques, as well as the computational effort involved in producing these architectures. This comparison demonstrates that GFAM does very well compared to a number of other approaches proposed in the literature that have claimed that they address the ART category proliferation problem.

3. The fuzzy ARTMAP architecture

Since Fuzzy ARTMAP's introduction (Carpenter et al., 1992), a number of Fuzzy ARTMAP variations, and associated successful applications of this ART family of networks have appeared in the literature.

The block diagram of FAM is shown in Fig. 1. Notice that this block diagram is simpler than the block diagram of FAM, reported in Carpenter and Grossberg in 1992, but very similar to the block diagram depicted in Kasuba (1993), as well as Taghi, Bagmisheh, and Pavesic (2003). Initially, Kasuba in 1993, and later on, other researchers, including Taghi et al. (2003), adopted this simpler FAM architecture because it is equivalent to the more complicated FAM architecture in Carpenter's and Grossberg's paper (1992), when FAM is confronted with classification problems. As the focus of our paper is on classification problems, we also adopt this simpler FAM architecture. The FAM architecture, depicted in 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 these input patterns are formed (designated as $\mathbf{w}_j^a$, and called templates), and the output layer (F_2^b) that holds the labels of the categories formed in the category representation layer. An

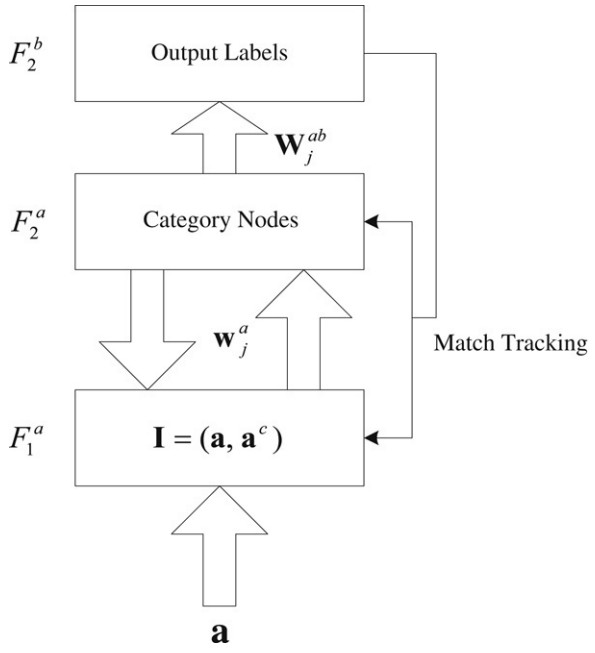

Fig. 1. Block-diagram of a Fuzzy ARTMAP architecture.

input pattern $\mathbf{I}$, presented at the input layer F_1^a of FAM, has the following form:

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1, a_2 \ldots, a_{M_a}, a_1^c, a_2^c, \ldots, a_{M_a}^c)$$

where,

$$a_i^c = 1 - a_i; \quad \forall i \in \{1, 2, \ldots, M_a\}.$$

The assumption here is that the input vector $\mathbf{a}$ is such that each one of its components lies in the interval [0, 1]. Any input pattern can be represented by the input vector $\mathbf{a}$, through an appropriate normalization, where M_a stands for the dimensionality of this input pattern. The above operation that creates an input vector $\mathbf{I}$ from the input vector $\mathbf{a}$ is called complementary encoding. Normalization of the components of the input patterns has been a common practice in the neural network literature. Complementary encoding of the input vector is necessary for the successful training of FAM (see Georgiopoulos, Huang, and Heileman (1994), where an example is provided for which, without complementary encoding of the input vector, FAM training fails).

FAM can operate in two distinct phases: the training phase and the performance (test) phase. The training phase of FAM can be described as follows: Given a set of inputs and associated label pairs, $\{(\mathbf{I}^1, \text{label}(\mathbf{I}^1)), \ldots, (\mathbf{I}^r, \text{label}(\mathbf{I}^r)), \ldots, (\mathbf{I}^{\text{PT}}, \text{label}(\mathbf{I}^{\text{PT}}))\}$ (called the training set), we want to train FAM to map every input pattern of the training set to its corresponding label. To achieve the aforementioned goal we present the training set to the FAM architecture repeatedly. That is, we present $\mathbf{I}^1$ to F_1^a, label($\mathbf{I}^1$) to F_2^b, $\mathbf{I}^2$ to F_1^a, label($\mathbf{I}^2$) to F_2^b, and finally, $\mathbf{I}^{\text{PT}}$ to F_1^a, label($\mathbf{I}^{\text{PT}}$) to F_2^b. We present the training set to Fuzzy ARTMAP as many times as necessary for FAM to correctly classify these input patterns. The task is considered accomplished (i.e., learning is complete) when the weights in

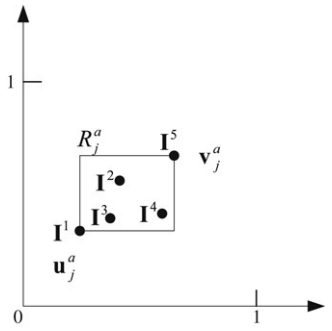

Fig. 2. A FAM category is represented by two vectors $\mathbf{u}_j^a$ and $\mathbf{v}_j^a$, this category has learned few.

FAM do not change during a training set presentation, or after a specific number of list presentations is reached. The performance phase of Fuzzy ARTMAP works as follows: Given a set of input patterns $\tilde{\mathbf{I}}^1, \tilde{\mathbf{I}}^2, \ldots, \tilde{\mathbf{I}}^{\text{PS}}$ (referred to as the test set), we want to find the Fuzzy ARTMAP output (label) produced when each one of the aforementioned test patterns is presented at its F_1^a layer. In order to achieve this goal, we present the test set to the trained Fuzzy ARTMAP architecture and we observe the network's output. For the purposes of this paper, we assume that the reader knows the details of the training/performance phases in Fuzzy ARTMAP.

What is worth emphasizing again is that the weights (templates) in Fuzzy ARTMAP create compressed representations of the input patterns presented to Fuzzy ARTMAP during its training phase. These compressed representations have a geometrical interpretation. In particular, every node (category) in the category representation layer of Fuzzy ARTMAP has template weights that completely define the lower and upper endpoints of a hyperbox. This hyperbox includes within its boundaries all the input patterns that chose this category as their representative category in FAM's training phase and were subsequently encoded by this category. In Fig. 2, we illustrate the hyperbox representation of a category j in FAM (in two dimensions) that has encoded input patterns $\mathbf{I}^1, \mathbf{I}^2, \mathbf{I}^3, \mathbf{I}^4$ and $\mathbf{I}^5$. A category j in FAM can be represented by its lower and upper endpoints as follows:

$$\mathbf{w}_j^a = (\mathbf{u}_j^a, \{\mathbf{v}_j^a\}^c).$$

The vector $\mathbf{u}_j^a$ is the vector with components the minimum values of the components of the input patterns (the $\mathbf{a}$'s) that were encoded by this category. Please note that $\{\mathbf{v}_j^a\}^c$ is the complement of $\mathbf{v}_j^a$. The vector $\mathbf{v}_j^a$ is the vector with components the maximum values of the components of the input patterns (the $\mathbf{a}$'s) that were encoded by this category. These vectors are defined below:

$$\mathbf{u}_j^a = \wedge_{i=1}^P \mathbf{a}(i) \quad \text{and} \quad \mathbf{v}_j^a = \vee_{i=1}^P \mathbf{a}(i).$$

In the above equation the symbol $\wedge$ ('fuzzy min") means that we take the minimum of all the components of the $\mathbf{a}(i)$'s to produce $\mathbf{u}_j^a$. Also in the above equation, the symbol $\vee$ ('fuzzy max") means that we take the maximum of all the components of the $\mathbf{a}(i)$'s to produce $\mathbf{v}_j^a$. In the above equations, P is the

maximum index of the input patterns that chose and were encoded by category j in FAM.

As can be seen from Fig. 2, the input patterns are simply represented by the first half of their components (i.e., corresponding vectors **a**). In Fig. 3, we show how a Fuzzy ARTMAP category is expanded (during FAM's training) to encode the information conveyed by a new input pattern $\mathbf{I}^*$ (with corresponding geometrical representation $\mathbf{a}^*$). In a similar fashion, Fig. 3 shows that the expansion of a category is not needed if the new input pattern $\mathbf{I}^*$ is contained within the boundaries of the hyperbox that represents this category. At the beginning, every category of FAM starts as a trivial hyperbox (equal to a point) and subsequently it expands, and its size is increased, to incorporate within its boundaries all the input patterns that in the training phase choose this hyperbox as their representative hyperbox, and are encoded by it. There are many ways of defining the size of a hyperbox, such as the Euclidean distance (L_2 distance between its lower and upper endpoints). In FAM the size of a hyperbox is defined to be the L_1 distance between its lower and upper endpoints, which ends up being the sum of the lengths of all the sides of the hyperbox. The size of a hyperbox, is allowed to expand up to a point allowed by a FAM network parameter denoted as the baseline vigilance parameter in FAM, and designated by the symbol $\bar{\rho}_a$. During the operation of FAM a vigilance ratio is computed, whose numerator is equal to (M_a—*size of the hyperbox*), and the denominator is equal to M_a. A hyperbox is deemed appropriate to encode an input pattern if this ratio is larger than or equal to FAM's vigilance parameter. Since this vigilance ratio is always a number between 0 and 1, it suffices to consider baseline vigilance parameter values only in the interval [0, 1]. Small values of this parameter allow the creation of large hyperboxes, while large values of this parameter allow the creation of small hyperboxes. In the one extreme when $\bar{\rho}_a$ is equal to 0, a hyperbox equal to the whole input space could be created, while at the other extreme when $\bar{\rho}_a$ is equal to 1 only point categories are formed. It turns out that this parameter has a significant effect on the number and type of categories (hyperboxes) formed in Fuzzy ARTMAP, and consequently it affects the performance of FAM. The performance of FAM is measured in terms of the number of categories created in its training phase (small number of categories is good), and how well it generalizes on unseen data (high generalization accuracy is good). In the process of discovering GFAM we are starting from a population of trained FAM networks that have been trained with different baseline vigilance parameter values, and as a result they attain different performances (in terms of size of the FAM networks created and in terms of the achieved generalization performance). It turns out that Fuzzy ARTMAP performance is also affected by the order in which patterns are presented to FAM in its training phase. In our experiments we used multiple differently trained FAM networks as members of the initial population in our evolutionary process because we changed the baseline vigilance, as well as the order of training pattern presentation for these FAM networks. The performance of FAM is also affected by another network parameter called choice parameter (denoted by the symbol β_a). In the training of the Fuzzy ARTMAP networks, used in our experiments, we fixed this choice parameter to the value of 0.1 (quite often used in Fuzzy ARTMAP training).

4. The evolution of FAM–GFAM

In the rest of the paper we assume that for every classification problem that we focus on we are provided with a training set, a validation test, and a test set. The training set is used for the training of GFAM architectures under consideration. The validation set is used to optimize the produced GFAM network in ways that will become apparent in the following text. Finally, the test set is used to assess the performance of the optimized GFAM network created.

GFAM (Genetic Fuzzy ARTMAP) is an evolved FAM network that is produced by applying, repeatedly, genetic operators on an initial population of trained FAM networks. GFAM uses tournament selection with elitism, as well as genetic operators, including crossover and mutation. In addition, GFAM uses a special operator, Cat_{del}; this special operator is needed so that categories could be destroyed in the ART network, thus leading us, through evolution, to smaller ART structures.

To better understand how GFAM is designed we resort to a step-by-step description of this design. It is instructive though to first introduce some terminology that is included in Appendix A. The design of GFAM can be articulated through a sequence of steps, defined succinctly below. The following pseudo-code shows the basic steps of GFAM:

```
Begin
1:  Generate-Initial-Population ()
2:  Repeat
2. a.    Evaluate-Fitness-And-Sort ()
2. b.    Selection ()
2. c.    Crossover ()
2. d.    Catdel ()
2. e.    Mutdate ()
   Until [a stopping criterion is met]
3.  Return bestNetwork
End
```

Step 1: *Generate initial population*: The algorithm starts by training Pop_{size} FAM networks, each one of them trained with a different value of the baseline vigilance parameter $\bar{\rho}_a$, and order of training pattern presentation. In particular, we first define $\bar{\rho}_a^{\text{inc}} = \frac{\bar{\rho}_a^{\max} - \bar{\rho}_a^{\min}}{\text{Pop}_{\text{size}} - 1}$, and then the baseline vigilance parameter of every network is determined by the equation $\bar{\rho}_a^{\min} + i * \bar{\rho}_a^{\text{inc}}$, where $i \in \{0, 1, \ldots, \text{Pop}_{\text{size}} - 1\}$. In our experiments with GFAM we chose $\bar{\rho}_a^{\min} = 0.1$, and $\bar{\rho}_a^{\max} = 0.95$. The choice parameter was chosen to be equal to 0.1.

We assume that the reader is familiar with how training of FAM networks is accomplished, and thus the details here are omitted. Once the Pop_{size} networks are trained, they need to be converted to chromosomes so that they can be manipulated by the genetic operators. GFAM uses a real number representation to encode the networks. Each FAM chromosome consists of two levels: level 1 containing all the categories of the FAM

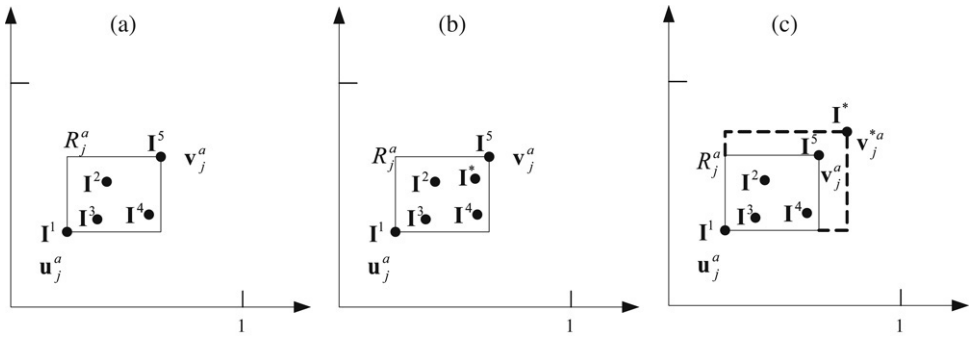

Fig. 3. FAM learning: a. A FAM category that learned 5 patterns. b. A new pattern $\mathbf{I}^*$ was presented inside the category, and thus the category does not expand. c. A new input pattern $\mathbf{I}^*$ was presented outside the category, and thus the category expands to include the input pattern $\mathbf{I}^*$.

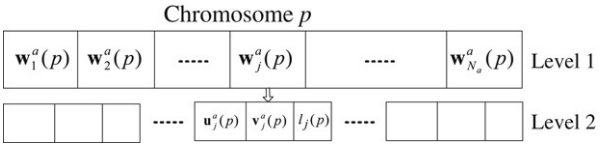

Fig. 4. GFAM chromosome structure.

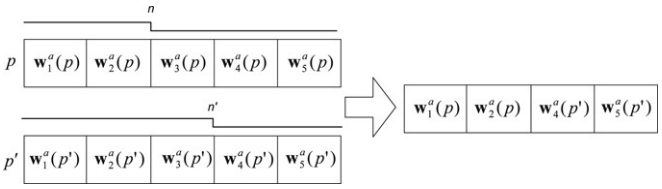

Fig. 5. Crossover implementation in GFAM.

network, and level 2 containing the lower and upper endpoints of every category in level 1, as well as the label of that category (see Fig. 4). We denote the category of a trained FAM network with index p ($1 \leq p \leq \text{Pop}_{\text{size}}$) by $\mathbf{w}_j^a(p)$, where $\mathbf{w}_j^a(p) = (\mathbf{u}_j^a(p), (\mathbf{v}_j^a(p))^c)$, where $\mathbf{u}_j^a(p)$ and $\mathbf{v}_j^a(p)$ are the lower and upper endpoints of the hyperbox corresponding to this category, and $l_j(p)$ is the label of this category.

Step 2: *Apply genetic operators*: In this step a GA is applied to the population of the ART trained networks.

Sub-step 2a: *Fitness evaluation*: Calculate the fitness of each ART chromosome (ART trained network). The fitness function for the p-th ART network is denoted by $\text{Fit}(p)$, and it depends on the $\text{PCC}(p)$ and $N_a(p)$ values of this network. Note that, $\text{PCC}(p)$designates the percentage of correct classification, exhibited by the p-th network, on the validation set, while $N_a(p)$ is the number of categories of the p-th network. The fitness function $\text{Fit}(p)$ of the p-th network is defined as follows:

$$\text{Fit}(p) = \text{PCC}(p) - \alpha(N_a(p) - \text{Cat}_{\min}). \quad (1)$$

Obviously, this fitness function increases as $\text{PCC}(p)$ increases or as $N_a(p)$ decreases. The value of $\text{Cat}_{\min}$ is chosen to be equal to the number of classes of the classification problem at hand. The parameter, α, in the fitness function equation above, is a user controlled parameter. It can be used to dictate the user's preference of accuracy versus complexity. It is evident from the fitness equation, that if we choose $\alpha = 0.5$, one percentage of better correct classification of a network, or two categories less of a network, increase the fitness function by the same amount (i.e., by an amount of 1); other values of α have similar interpretations. This is one of the simplest ways of defining a fitness function that depends on two measures (generalization performance of the network and size of the network) and has been extensively adopted in the classification literature (e.g., see Breiman, Friedman, Olshen, and Stone (1984), Palmes, Hayasaka, and Usui (2005)).

Sub-step 2b: *Selection*: Initialize an empty generation referred to as the temporary generation. The algorithm searches for the best NC_{best} chromosomes (i.e., the chromosomes having the NC_{best} highest fitness values) from the current generation and copies them to the temporary generation without change.

Sub-step 2c: *Cross-over operation*: The remaining Pop_{size}–NC_{best} chromosomes in the temporary generation are created by crossing over pairs of parents from the current generation. The parents are chosen using a deterministic tournament selection, as follows: Randomly select two groups of four chromosomes each from the current generation, and use as a parent, from each group, the chromosome with the best fitness value in the group. If it happens that from both groups the same chromosome is chosen then we choose from one of the groups the chromosome with the second best fitness value. If two parents with indices p, p' are crossed over two random numbers n, n' are generated from the index sets $\{1, 2, \ldots, N_a(p)\}$ and $\{1, 2, \ldots, N_a(p')\}$, respectively. Then, all the categories with index greater than index n' in the chromosome with index p' and all the categories with index less than index n in the chromosome with index p are moved into an empty chromosome within the temporary generation. Notice that crossover is done on level 1 of the chromosome. This operation is pictorially illustrated in Fig. 5.

Sub-step 2d: *Category deletion*: The operator Cat_{del} deletes one of the categories of every chromosome (created in step 2c) with probability $P(\text{Cat}_{\text{del}})$. If a chromosome is chosen to have one of its categories deleted, then this category is picked randomly from the collection of the chromosome's categories; however deletion is prohibited if it would violate the class inclusion criterion. The class inclusion criterion dictates that, in every network, there is at least one category for each class label present in the data.

Sub-step 2e: *Category mutation*: With probability $P(\text{mut})$ every chromosome created by step 2d gets mutated as follows:

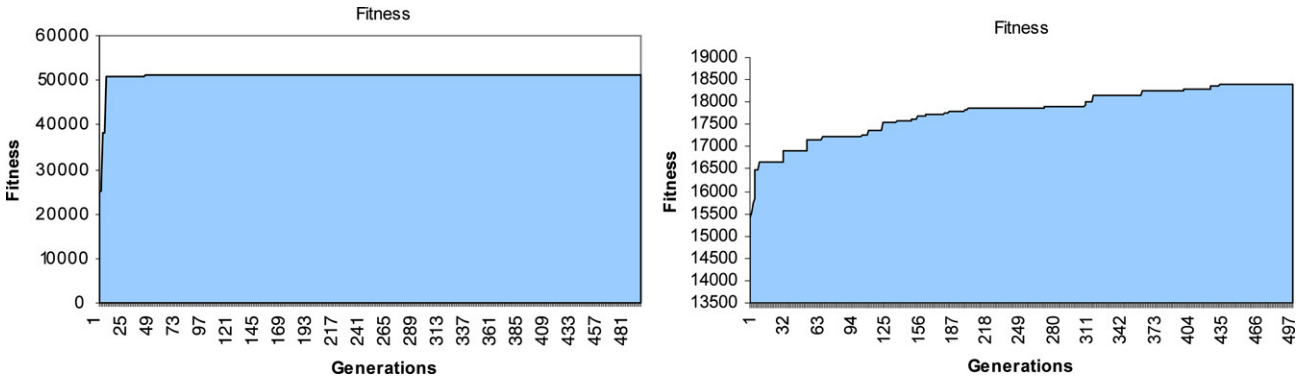

Fig. 6. The two graphs were obtained by plotting the fitness function value of GFAM against the generation number. Each experiment was run for 500 generations. The first problem is a relatively easy problem for GFAM. The second problem is harder. Examining these graphs reveals a number of issues that need to be considered when implementing the stopping criteria. The left graph shows that maximum performance is achieved early in the process (at about generation 25). There was no need to continue the evolution of FAMs for such a large number of generations (500). The graph on the right shows the change in fitness function, as generations progress, for a harder database. It can be observed that for many consecutive periods no appreciable change in the fitness is observed. In this case, a stopping criterion that depends on the improvement might be triggered prematurely resulting in poor results. A stopping criterion should wait for an appropriate window of no-improvement before it is triggered.

For each category, either its **u** or **v** endpoint is selected randomly (50% probability) and then every component of this selected vector gets mutated by adding to it a small number. This number is drawn from a Gaussian distribution with zero mean and standard deviation of 0.01. If the component of the chosen vector becomes smaller than 0 or greater than 1 (after mutation), it is set back to 0 or 1, respectively. Notice that mutation is applied at level 2 of the chromosome structure. The label of the chromosome is not mutated because our initial GA population consists of trained FAMs, and consequently we have a lot of confidence in the labels of the categories that these trained FAMs have discovered through the FAM training process.

Step 3: *Check stopping criteria*: If a stopping criterion is not met, replace the current generation with the members of the temporary generation and go to step 2a. Otherwise, terminate and return the best network.

There is a need for an automated stopping criterion so that the evolution of FAMs does not proceed for unnecessarily many generations. Ideally, the evolution should be allowed to proceed for as long as necessary, and it should terminate when network performance improvements are not attainable any more. In practice though, there is a trade-off between network performance improvements and computational effort expended to achieve these improvements.

It might be beneficial to use multiple stopping criteria to terminate the evolution of FAM networks. One obvious stopping criterion is to set a threshold for the maximum number of generations, $\text{Gen}_{\max}$, that the evolution of FAMs is allowed to continue. The advantage of having this stopping criterion is that it ensures that the algorithms will always terminate and would not get trapped in an infinite loop if the other stopping criteria are never triggered. The user can always set the maximum number of iterations to a large number to allow the algorithm to terminate using other, more appropriate, stopping criteria.

Another popular stopping criterion is to stop when no more improvement in fitness is observed. To ensure the lack of improvement is not due to the stochasticity of the search, the evolution of FAMs is terminated only when no significant network performance improvements are observed for a number of consecutive evolutions. This number of consecutive evolutions can be chosen to be a percentage of the maximum number of generations $\text{Gen}_{\max}$; selecting this number correctly is a difficult feat (see Fig. 6). In our experiments we chose $\text{Gen}_{\max} = 500$, and furthermore we stopped the evolution of FAMs if 50 generations (10% of $\text{Gen}_{\max}$) elapsed without an appreciable network fitness improvement. Appreciable network fitness improvement is an improvement larger than 0.01.

5. Experiments with GFAM

We conducted a number of experiments to assess the performance of the genetically engineered Fuzzy ARTMAP (GFAM) architecture. There were two objectives for this experimentation.

- The first objective is to find good (default) values for the ranges of two of the GA parameters, the probability of deleting a category, $P(\text{Cat}_{\text{del}})$, and the probability of mutating a category, $P(\text{mut})$. The default values were identified by conducting experiments with 19 databases. This effort is described in detail in Section 5.2.
- The second objective is to compare the GFAM performance (for the default parameter values) to that of popular ART architectures that have been proposed in the literature with the intent of addressing the category proliferation problem in FAM, such as ssFAM, ssEAM, ssGAM, and micro-ARTMAP. This effort is described in Section 5.3.

5.1. Databases

We experimented with both artificial and real databases. Table 1 shows the specifics of these databases.

Gaussian databases (Database index 1–12): These are artificial databases, where we created 2-dimensional data sets,

Table 1
Databases used in the GFAM, ART experiments

	Database name	# Training instances	# Validation instances	# Test instances	# Attributes	# Classes	% Major class
1	G2c-05	500	5000	5000	2	2	50.00
2	G2c-15	500	5000	5000	2	2	50.00
3	G2c-25	500	5000	5000	2	2	50.00
4	G2c-40	500	5000	5000	2	2	50.00
5	G4c-05	500	5000	5000	2	4	25.00
6	G4c-15	500	5000	5000	2	4	25.00
7	G4c-25	500	5000	5000	2	4	25.00
8	G4c-40	500	5000	5000	2	4	25.00
9	G6c-05	504	5004	5004	2	6	16.67
10	G6c-15	504	5004	5004	2	6	16.67
11	G6c-25	504	5004	5004	2	6	16.67
12	G6c-40	504	5004	5004	2	6	16.67
13	4Ci/Sq	2000	5000	3000	2	5	20.00
14	1Ci/Sq	2000	5000	3000	2	2	50.00
15	1Ci/Sq/70:30	2000	5000	3000	2	2	70.00
16	2Ci/Sq/50:30:20	2000	5000	3000	2	3	50.00
17	MOD-IRIS	500	4800	4800	2	2	50.00
18	ABALONE	501	1838	1838	7	3	33.30
19	PAGE	500	2486	2487	10	5	83.20
20	OPTDIGITS	1823	2000	1797	64	10	10.00
21	PENDIGITS	4494	3000	3498	16	10	10.00
22	SAT	2000	2436	2000	36	6	24.19
23	SEG	800	810	700	19	7	14.29
24	WAV	1000	2000	2000	21	3	33.33
25	SHUTTLE	3000	1000	54,000	9	5	80.00
26	GLASS	75	75	64	9	6	35.51
27	PIMA	150	150	232	7	2	65.10

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 a 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.

Structures within a structure databases (Database index 13–16): These are artificial databases that were inspired by the circle (structure) – in the – square (structure) problem. This problem has been extensively examined in the ART, and other than ART neural network literature. Eight different datasets were generated by changing the structures (type, number and probability) that we were dealing with. The data-points within each structure of these artificial datasets are uniformly distributed within the structure. The number of points within each structure is chosen in a way that the probability of finding a point within this structure is equal to a pre-specified number. Some of these artificial datasets were also considered in the Parrado-Hernandez et al. (2003) paper where four different ART architectures were compared, Fuzzy ARTMAP, FasART, distributed Fuzzy ARTMAP, and distributed FasART.

(a) 4Ci/Sq: This is a four circle in a square problem, a five class classification problem. The probability of finding a data point within a circle or inside the square and outside the circles is equal to 0.2.
(b) 1Ci/Sq: This is a one circle in a square problem, a two class classification problem. The probability of finding a data point within a circle or inside the square and outside the circle is equal to 1/2. The sizes of the areas in the circle and outside the circle and inside the square are the same. This is the benchmark circle in the square problem.
(c) 1Ci/Sq/30:70: This is a one circle in a square problem, a two class classification problem. The probability of finding a data point within a circle or inside the square and outside the circle is equal to 0.3 and 0.7, respectively. The sizes of the areas in the circle and outside the circle and inside the square are 0.3 and 0.7, respectively. This is a modified version of the circle in the square problem.
(d) 2Ci/Sq/20:30:50: This is two circles in a square problem, a three class classification problem. One of the circles is smaller than the other. The probabilities of finding a data point within the small circle, the large circle, and outside the circles but inside the square are 0.2, 0.3, and 0.5, respectively.

In Figs. 7 and 8 we show plots of the simulated databases.

Modified iris database (MOD-IRIS) (Database index 17): In this database we started from the IRIS dataset (Newman, Hettich, Blake, & Merz, 1998) of the 150 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

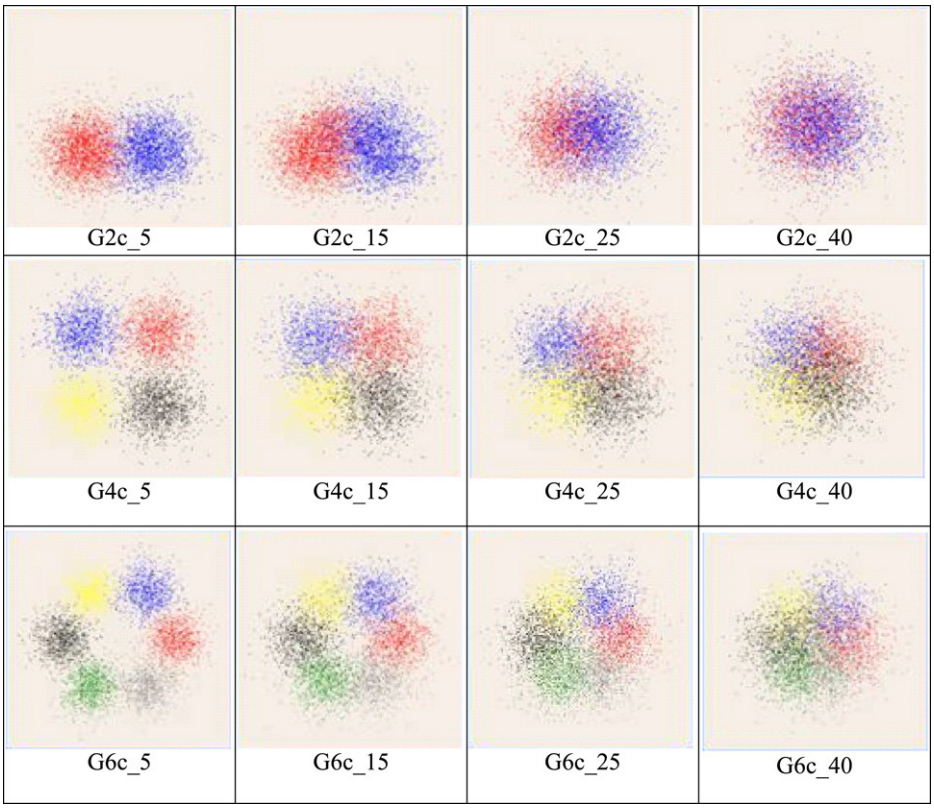

Fig. 7. Gaussian databases (2-dimensional, 2, 4 or 6 class, 5%, 15%, 25% and 40% of overlap).

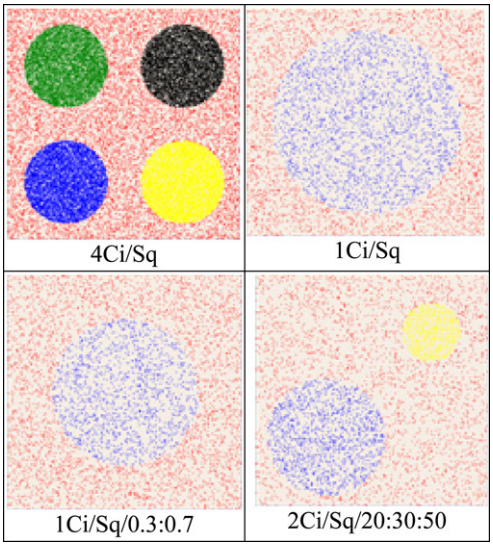

Fig. 8. Structures within structure databases.

they seem to have enough discriminatory power to separate the 2-class data. Finally, in order to create a reasonable size dataset from these 100 points (so we can reliably perform cross-validation to identify the optimal ART, GFAM networks) 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 approximately 10,000 points. We named this database Modified Iris.

Modified abalone database (ABALONE) (Database index 18): This database is originally used for prediction of the age of an abalone (Newman et al., 1998). It contains 4177 instances, each with 7 numerical attributes, 1 categorical attribute, and 1 numerical target output (age). We discarded the categorical attribute in our experiments, and grouped the target output values into 3 classes: 8 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.

Page blocks database (PAGE) (Database index 19): This database represents the problem of classifying the blocks of the page layout in a document (Newman et al., 1998). It contains 5473 examples coming from 54 distinct documents. Each example has 10 numerical attributes (e.g., height of the block, length of the block, eccentricity of the block, etc.,) and one target (output) attribute, representing the type of the block (text, horizontal line, graphic, vertical line, and picture). One of the noteworthy points about this database is that its major class (text) has a high probability of occurring (about 90%). This dataset has five classes; four of them make only 10% of the total instances.

Optdigits (OPT) (Database index 20): This UCI (Newman et al., 1998) database has vectors representing normalized bitmaps of handwritten digits from a preprinted form. The bitmaps were normalized using preprocessing programs made available to UCI by NIST. From a total of 43 people, 30 contributed to the training set and the remaining 13 to the test set. 32×32 bitmaps are divided into non-overlapping blocks of 4×4 and the number of pixels is counted in each block. This generates an input matrix of 8×8 where each element is an integer in the range 0–16. This database has 64 attributes and 10 classes. The

training set has 3823 records and the test set has 1797 records. In our experiments, we divided the original training set into a training set of 1823 records and a validation set of 2000 records.

Pendigits (PEN) (Database index 21): This UCI (Newman et al., 1998) database has records representing hand-written digits. The database was created by collecting 250 samples from 44 writers. The samples written by 30 writers are used for training, cross-validation and writer dependent testing, and the digits written by the remaining 14 writers are used for writer independent testing. This database has 16 attributes and 10 classes. The training set has 7494 records and the test set has 3498 records. We divided the original training set into a training set of 4494 records and a validation set of 3000 records.

Satellite image (SAT) (Database index 22): This UCI (Newman et al., 1998) dataset gives the multi-spectral values of pixels within 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in each neighborhood. The aim is to predict the classification given the multi-spectral values. There are six classes and thirty-six numerical attributes. The training set consists of 4435 records while the test set consists of 2000 records. The original training set was divided into a training set of 2000 records and a validation set of 2435.

Image segmentation (SEG) (Database index 23): This UCI (Newman et al., 1998) dataset was used in the StatLog Project. The samples are from a database of seven outdoor images. The images are hand-segmented to create a classification for every pixel as one of brickface, sky, foliage, cement, window, path, or grass. There are seven classes, nineteen numerical attributes and 2310 records in the dataset.

Waveform (WAV) (Database index 24): This is an artificial three-class problem based on three wave-forms (Newman et al., 1998). Each class consists of a random convex combination of two waveforms sampled at the integers with noise added. A description for generating the data is given in Breiman et al. (1984) and a C program is available from UCI. There are twenty-one numerical attributes, and 3000 records in the training set. Error rates are estimated from an independent test set of 2000 records. The original training set was divided into a training set of 1000 records and a validation set of 2000 records.

Shuttle (SHU) (Database index 25): This UCI (Newman et al., 1998) database contains 9 attributes all of which are numerical and five classes. Approximately 80% of the data belongs to one class. The training set has 43,500 records and test set has 14,500 records. In our experiments we used 3000 records from the original training set for training and 1000 for validation. The rest were added to the test set.

Glass (GLS) (Database index 26): This UCI (Newman et al., 1998) database is used to classify types of glass. It was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence, if it is correctly identified. This database has 214 instances, 10 numerical attributes and 6 classes.

Pima-Indian diabetes (PIM) (Database index 27): This UCI (Newman et al., 1998) database was contributed by V. Sigillito. The patients in the dataset are females at least twenty-one years old of Pima Indian heritage living near Phoenix, Arizona, USA. The problem is to predict whether a patient would test positive for diabetes given a number of physiological measurements and medical test results. There are 2 classes, 8 numerical attributes, and 768 records. However, many of the attributes, notably serum insulin, contain zero values which are physically impossible. We remove serum insulin and records that have impossible values in other attributes, resulting in 7 attributes and 532 records (this approach is followed by other researchers).

The summarized specifics of each one of these databases are depicted in Table 1. Please note that the first 19 databases are used for objectives 1 and 2.

5.2. Selection of the GA parameters

As we have mentioned above, the first objective of our experimentation was devoted to the selection of good values for the GA parameters: probability of deleting a FAM category, $P(\text{Cat}_{\text{del}})$, and probability of mutating a FAM category, $P(\text{mut})$. As is evident from our prior discussion, there are a few other GA parameters that one has to carefully choose, such as Pop_{size}, Gen_{max}, and NC_{best}; we did not perform exhaustive experimentation to decide on the values of these parameters, but limited experimentation with these parameters for some of the above databases showed that reasonable choices for these parameters were: $\text{Pop}_{\text{size}} = 20$, $\text{Gen}_{\text{max}} = 500$, and $\text{NC}_{\text{best}} = 3$.

Our approach to select good values for the GA parameters $P(\text{Cat}_{\text{del}})$ and $P(\text{mut})$ consisted of a number of steps delineated below:

Select GA step 1: We selected four different values for the $P(\text{Cat}_{\text{del}})$ to experiment with. These were: 0.05, 0.1, 0.2, and 0.4. We also selected four different values for the $P(\text{mut})$ to experiment with. These were: 0.0, 0.1, 0.2, and 0.4. This resulted in 16 combinations of parameter settings for $P(\text{Cat}_{\text{del}})$, and $P(\text{mut})$.

Select GA step 2: For each one of the 16 settings of the $P(\text{Cat}_{\text{del}})$, and $P(\text{mut})$ parameters, and for each of the 19 databases listed in Table 2, and described in an earlier section, we applied the GA optimization of FAMs, as delineated in Section 4, 10 different times (using a different initial seed for the GA optimization). Consequently, for each database, and each parameter setting, we obtained 10 PCC (accuracy) and 10 N_a (size) numbers.

Select GA step 3: For each dataset, we chose the best-performing (with respect to average validation PCC of the 10 experiments) parameter setting. Then, we used ANOVA statistical tests to choose other parameter settings that did not significantly differ (statistically) from the best performing parameter setting. We marked these parameter settings as "good" settings for this database.

Select GA step 4: After we performed step 3 for all databases we counted the number of databases for which a particular parameter setting was deemed as "good" from the Select GA step 3. Based on these counts we recommended the best parameter setting for each GFAM algorithm, and a range of acceptable parameter settings.

Table 2
Goodness of parameter settings for the GA parameters $P(\mathrm{Cat}_{\mathrm{del}})$, and $P(\mathrm{mut})$

Description	Data	g2c_05	g2c_15	g2c_25	g2c_40	g4c_05	g4c_15	g4c_25	g4c_40	g6c_05	g6c_15	g6c_25	g6c_40	4cirinsq	cirInSq	cins_70_30	2cins_50_30_20	Iris	Abalon	Pageblocks	Sum	Best
pDel = 0.05, pMutate = 0.0	PCC	1																			1	
	Size																					
pDel = 0.05, pMutate = 0.1	PCC	1	1	1	1	1	1	1	1	1	1	1		1		1	1	1	1	1	17	4
	Size																					
pDel = 0.05, pMutate = 0.2	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	
	Size																			*		
pDel = 0.05, pMutate = 0.4	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
	Size																					
pDel = 0.1, pMutate = 0.0	PCC																					
	Size																					
pDel = 0.1, pMutate = 0.1	PCC	1	1	1	1	1	1	1		1	1	1		1		1	1	1	1	1	16	
	Size																					
pDel = 0.1, pMutate = 0.2	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1
	Size																					
pDel = 0.1, pMutate = 0.4	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	7
	Size																*		*			
pDel = 0.2, pMutate = 0.0	PCC	1																			1	
	Size																					
pDel = 0.2, pMutate = 0.1	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1		1	17	1
	Size																					
pDel = 0.2, pMutate = 0.2	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	2
	Size														*							
pDel = 0.2, pMutate = 0.4	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1
	Size																					
pDel = 0.4, pMutate = 0.0	PCC																				0	
	Size																					
pDel = 0.4, pMutate = 0.1	PCC	1	1	1	1	1	1	1	1	1	1	1		1	1		1	1	1	1	17	2
	Size															*						
pDel = 0.4, pMutate = 0.2	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1
	Size																					
pDel = 0.4, pMutate = 0.4	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	2
	Size													*								

An entry of "1" in the table indicates that the corresponding parameter setting is good, while the lack of an entry "1" indicates that the parameter setting is not good. The column before last counts the number of "1" entries for a particular parameter setting that are good, while the last column counts the number of times that a particular parameter setting is the best of all parameter settings (with respect to the average PCC on the validation set or the average number of categories). Entries are depicted for the PCC value and not for the number of categories value, in order to avoid cluttering the table; however whenever there is a "1" or lack of a "1" entry for the PCC value there is also a "1" or lack of a "1" entry for the number of categories value. Underscored "1" entries in this table designate parameter settings for which we obtained the best result related to the average PCC value, while asterisks designate parameter settings for which we obtained the best result with respect to the average number of categories value.

Table 2 summarizes the results. In Table 2 an entry of "1" for a database indicates that the corresponding parameter setting performed well (with respect to the average PCC on the validation set). An underscored "1" entry indicates that the corresponding parameter setting performed the best for this database (with respect to the average PCC on the validation set). In Table 2, the "1" entries corresponding to the Number of Categories criterion (actually average number of categories criterion) are omitted to preserve the table's clarity. In Table 2 we designated with an asterisk the parameter setting that performed best for this database with respect to the average Number of Categories criterion. A careful observation of the results shown in Table 2 indicate that any value of $P(\text{Cat}_{\text{del}})$ in the interval [0.2, 0.4], and any value of the $P(\text{mut})$ in the interval [0.05, 0.4] gives good results. Also, the results from Table 2 indicate that the best performing parameter setting for GFAM is $P(\text{Cat}_{\text{del}}) = 0.1$, and $P(\text{mut}) = 0.4$, since for this parameter setting we observe the highest number of good performances (19), and best performances (7) of the associated GFAM (the count of the best performances consider the best observed performances with respect to the average PCC on the validation set or the average number of categories). Finally, we can also deduce from the results of Table 2 that a probability of mutation equal to 0 is not recommended, since it always (for most databases) results in a GFAM network that is not performing well.

5.3. Comparison of GFAM with other ART architectures

The second objective of our experimentation was to compare GFAM with other ART architectures that have previously appeared in the literature and addressed the category proliferation problem in ART.

The ART architectures that we chose to compare GFAM with are: ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP. The first three are based on the principle of semi-supervision, introduced by Anagnostopoulos et al. (2002), and Verzi et al. (2001). Semi-supervision is a term attributed to learning in ART architecture (FAM, EAM (Anagnostopoulos, 2001; Anagnostopoulos & Georgiopoulos, 2001) or GAM (Williamson, 1996)), where categories in ART are allowed to encode patterns of different labels provided that the percentage of patterns that belong to the plurality label exceed a certain threshold. Safe micro-ARTMAP is a Fuzzy ARTMAP that allows categories in Fuzzy ARTMAP to encode patterns that are mapped to different labels. In safe micro-ARTMAP (see Gomez-Sanchez et al. (2001)) though the mixture of labels allowed in a category, or in all of the categories, is controlled by the entropy of the category or categories.

As we have mentioned in an earlier section, in every classification problem that we focus on we are provided with a training set, a validation test, and a test set. The training set is used for the training of GFAM (ART) architectures under consideration. The validation set in the GFAM case is used to guide the evolution of the trained FAM networks from generation 1 to generation Gen_{max}. The validation set in the other ART networks' case is used to choose optimal values for the ART network parameters (e.g., vigilance, choice parameter, order of pattern presentation, etc); optimal values of ART network parameters are the ones that give the highest value of the already defined fitness function. The test set is used to assess the performance of the optimized GFAM (ART) network. The percentages of different class data-points in the training, validation and test set are the same as the ones found in the original dataset.

In the previous section, we have experimented extensively with GFAM to identify a good initialization of the GA process and to specify a good set of parameters for the evolution of trained FAMs. From this point on, the GFAM is produced by first initializing a population of 20 trained FAM networks (they were trained with different values of the baseline vigilance parameter and different orders of training pattern presentations), and by evolving them for a maximum of 500 generations. In particular, the GA parameters used for the creation of GFAM were: $\bar{\rho}_a^{\text{min}} = 0.1$, $\bar{\rho}_a^{\text{max}} = 0.95$, $\beta_a = 0.1$, $\text{Pop}_{\text{size}} = 20$, $\text{Gen}_{\text{max}} = 500$, $\text{NC}_{\text{best}} = 3$, $P(\text{Cat}_{\text{del}}) = 0.1$, $P(\text{mut}) = 0.4$. GFAM is the FAM network that attains the highest value of the fitness function at end of the evolutionary process.

For each of the ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP networks, and for each of the 19 databases, we performed a number of experiments with different settings of their network parameter values. For each one of these network parameter settings we calculated the resulting network's fitness function (we used the same fitness function as the one utilized for the GFAM networks (see Eq. (1))). For the training of ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP we used the same training set as the one used for the GFAM networks, and for the validation of the performance of each of the ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP networks we used the same validation set as the one used for the GFAM networks. The parameter setting of the ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP network that maximized the fitness function was chosen as the best parameter setting for the specific database; the number of categories created by the "best" parameter setting network, and its corresponding percentage of correct classification on the test set are reported in Table 3.

In particular, the parameter settings that we experimented with for ssFAM were: baseline vigilance values ranging from 0 to 0.9 with step size of 0.1, choice parameter values of 0.001 and 0.01, maximum allowable mixture threshold values ranging from 0 to 1 with step size of 0.1, and 100 different orders of pattern presentations of the training data (resulting in 22,000 different parameter settings). Furthermore, the settings for ssEAM were: baseline vigilance values ranging from 0 to 0.9 with step size of 0.1, choice parameter values of 0.001 and 0.01, maximum allowable mixture threshold values ranging from 0 to 1 with step size of 0.1, minimum axes to maximum axis ratio values ranging from 0.1 to 1 with step size of 0.1, and 100 different orders of pattern presentations of the training data (resulting in 220,000 different parameter settings). Also, the settings for ssGAM were: baseline vigilance values ranging from 0 to 0.9 with step size of 0.1, initial standard deviation

Table 3
Best results obtained from GFAM compared to best results obtained from safe μARTMAP, ssFAM, ssEAM and ssGAM

	Database name	GFAM		Safe μARTMAP		ssFAM		ssEAM		ssGAM	
		PCC	Size	PCC	Size	PCC	Size	PCC	Size	PCC	Size
1	G2c-05	95.40	2	95.22	2	94.70	2	94.68	2	94.66	2
2	G2c-15	85.32	2	85.00	2	85.58	2	85.26	2	85.50	2
3	G2c-25	75.24	2	74.98	2	75.08	2	75.16	2	75.06	2
4	G2c-40	62.02	2	61.40	3	59.56	2	59.50	2	59.56	3
5	G4c-05	95.16	4	95.04	4	94.54	5	94.92	4	95.56	4
6	G4c-15	84.78	4	83.28	4	82.74	4	82.06	4	83.42	6
7	G4c-25	75.08	4	74.50	4	70.38	9	72.92	4	72.32	21
8	G4c-40	59.94	4	58.90	4	57.80	7	54.70	7	59.54	14
9	G6c-05	94.84	6	92.36	6	87.23	8	93.46	6	94.66	8
10	G6c-15	84.89	6	80.91	6	80.59	6	82.03	7	83.49	9
11	G6c-25	74.36	6	67.92	6	70.24	15	71.44	7	71.24	13
12	G6c-40	60.13	6	54.07	6	55.15	17	49.30	7	55.15	13
13	4Ci/Sq	95.03	9	95.42	8	87.23	18	94.68	5	93.40	12
14	1Ci/Sq	97.77	7	94.76	8	92.97	8	97.02	8	91.02	8
15	1Ci/Sq/0.3:0.7	97.93	6	96.82	8	93.21	8	97.13	8	92.33	8
16	2Ci/Sq/20:30:50	97.57	5	97.22	6	90.24	12	97.01	3	95.60	9
17	MOD-IRIS	95.35	2	94.91	2	94.79	8	94.77	2	94.70	2
18	ABALONE	61.84	3	58.10	3	60.01	6	58.81	3	56.31	2
19	PAGE	96.77	5	92.92	5	87.93	3	93.84	2	94.32	5

parameter ranging from 0.1 to 1 with step size of 0.1, maximum allowable mixture threshold values ranging from 0 to 1 with step size of 0.1, and 100 different orders of pattern presentations of the training data (resulting in 110,000 different parameter settings). Finally, the settings for safe micro-ARTMAP were: baseline vigilance values ranging from 0 to 0.4 with step size of 0.2, baseline vigilance parameter values of 0.001 and 0.01, 5 values for the maximum "all-categories" entropy threshold, 6 different ratios of the values of the "categories" entropy threshold to the "all-categories" entropy threshold, three values of the maximum allowable expansion of a category, and 100 different orders of pattern presentations of the training data (resulting in 90,000 different parameter settings).

The best parameter setting, identified in the previous subsection for GFAM was used for each of the 19 databases. Ten (10) experiments per database were conducted for 10 different initial seeds of the GA optimization process. The network that produced the maximum value of the fitness function was deemed as "best". The number of categories of the "best" GFAM for each database and its corresponding performance (PCC) on the test set are reported in Table 3.

Some of the conclusions that can be deduced from the comparative results, depicted in Table 3, are emphasized below.

GFAM attains good performance on all the 19 datasets, and quite often, optimal performance. GFAM's performance on databases 1–12 (Gaussian datasets of known amount of overlap) is optimal, for all practical purposes; for example the best performance on the G6c-15 problem (6 class Gaussian dataset of 15% overlap) is a classifier with 6 categories and 85% correct classification, and GFAM is a classifier with six categories and 84.89% of correct classification. Finally, two of the real problems reported here, MOD-IRIS and PAGE, also gave very good results of 95.35% and 96.77% of correct classification, by creating the optimum number of two and five categories, respectively.

According to the results in Table 3, in all instances the accuracy of GFAM on a test set (generalization performance) is either higher than or practically equal to the accuracy of the other ART networks (ssFAM, ssEAM, ssGAM or safe micro-ARTMAP). Also, in most instances, the size of GFAM is smaller than the size of the other ART networks. For instance, ssFAM performs as well as the GFAM network for the 2-class Gaussian datasets. For all the other datasets the GFAM network performs better (achieving higher PCC with fewer ART categories). The largest difference in PCC observed is almost 8% (for the 4 Circle in the Square problem), while the largest ratio of number of ssFAM versus GFAM categories is for the modified IRIS problem (ratio of 4). ssEAM performs as well as the GFAM for the 2-class Gaussian datasets. For all the other datasets GFAM performs better (achieving higher PCC). The biggest difference in PCC is for the 6-class Gaussian, 40% overlap dataset, where GFAM achieves more than 10% better classification accuracy. In some problems GFAM and ssEAM created the same number of categories. In other problems (e.g. Gaussianly distributed datasets) GFAM created fewer categories. In the remaining problems, especially problems for which the data favor the ellipsoidal structure of the EAM categories, ssEAM created fewer categories than GFAM. ssGAM performs as well as the GFAM networks for the 2-class Gaussian datasets. For all the other datasets GFAM performs better (achieving higher PCC with fewer ART categories). The largest difference in PCC observed is more than 6% (for the 1 Circle in the Square problem), while the largest ratio of number of ssGAM versus GFAM categories is for the four Gaussian dataset with 25% overlap problem (ratio larger than 5). safe micro-ARTMAP performs as well as the GFAM network for the 2-class, and 4-class Gaussian datasets. For all the other datasets GFAM performs better (achieving higher PCC with fewer ART categories). The largest difference in PCC observed is more than 6% (for the 6 class Gaussian dataset with 25%

overlap). On the number of categories it can be observed that GFAM performed, as good as, or slightly better than, safe micro-ARTMAP on most databases.

What is also worth pointing out is that the better performance of the GFAM network is attained with reduced computations as compared with the computations needed by the alternate methods (ssFAM, ssEAM, ssGAM, safe micro-ARTMAP). Specifically, the performance attained by ssFAM, ssEAM, ssGAM and the safe micro-ARTMAP required training these networks for a large number of network parameter settings (at least 22,000 experiments) and then choosing the network that achieved the highest value for the fitness function that we introduced earlier (through cross-validation). In GFAM cases we trained only a small number of these networks ($Pop_{size} = 20$ of them), compared to the large number of networks trained in the ssFAM, ssEAM, ssGAM or micro-ARTMAP cases (at least 22,000). Furthermore, in GFAM we evolved the trained networks at most $Gen_{max} = 500$ times, each evolution requiring cross-validating $Pop_{size} = 20$ networks. Hence, the total number of networks cross-validated in the ssFAM, ssEAM, ssGAM and micro-ARTMAP cases were at least 22,000, while in the GFAM networks were 10,000; furthermore the networks cross-validated in the ssFAM, ssEAM, ssGAM, and micro-ARTMAP cases have higher number of category nodes than the ones cross-validated in the GFAM case. As a result, we can argue that the improved performance (in most instances, smaller number of categories and better generalization) of GFAM, compared with ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP, is achieved with reduced computational effort. Of course, one can claim that such an extensive experimentation with these networks might not be needed, especially if one is familiar with the functionality of these networks and chooses to experiment only with a limited set of network parameter values. However, the practitioner in the field might lack the expertise to carefully choose the network parameters to experiment with, and consequently might need to experiment extensively to come up with a good network. In this case, GFAM has an advantage over the other ART network approaches because it has already provided a list of default parameter settings for the evolution of trained FAMs, and as a result the experimentation with a separate validation set is not needed. In Appendix B, we show, in more detail, how the more computationally efficient GFAM compares to ssFAM, ssEAM, ssGAM and safe micro-ARTMAP. The comparison is based on the assumption that extensive parameter experimentation with the network parameters of ssFAM, ssEAM, ssGAM or safe micro-ARTMAP is needed to obtain a good performing ssFAM, ssEAM, ssGAM or safe micro-ARTMAP network, respectively.

The comparison of GFAM, and ssFAM, ssEAM, ssGAM, provided above, is fair because it used the same databases and datasets/per database for training, validation and testing of these architectures, as well as the same criterion for finding the best of these ART architectures (the criterion was to maximize the fitness function, defined in Section 4). However, some of the structures-in-a-structure artificial databases, extensively examined above, have also been utilized to assess the performance of other ART architectures, such as the distributed Fuzzy ARTMAP (dFAM), FasART, and distributed FasART (see Parrado-Hernandez et al. (2003)). Distributed Fuzzy ARTMAP differs from Fuzzy ARTMAP in the sense that more than one category is activated to represent an input pattern in ART's training phase. FasART uses a different activation function compared to the one used by Fuzzy ARTMAP. Finally, distributed FasART is the distributed version of FasART, in a similar manner as distributed Fuzzy ARTMAP is the distributed version of Fuzzy ARTMAP. More details about the functionality of these ART networks can be found in Parrado-Hernandez et al. (2003) and they are beyond the scope of this paper. We avoid the extensive comparison of GFAM with dFAM, FasART, and dFasART for a reason. Although some of the databases used to assess the performance of dFAM, FasART, and dFasART, in Parrado-Hernandez et al. (2003), are approximately the same as the databases used to assess the performance of GFAM, the actual data used for training, and testing of GFAM are not the same used for the training and testing of dFAM, FasART, and dFasART. Furthermore, parameter network optimization with a validation set, such as to optimize a fitness function, was not conducted for FasART, dFAM, and dFasART. Actually, the results reported in Parrado-Hernandez et al. (2003), are averages of the performances of the dFAM, FasART, and dFasART on a test set of 5000 points for a specific set of network parameter values (we tend to believe that it was a good set of network parameter values). The averages correspond to the average performance attained by 100 different choices of training sets of size equal to 2000 points. The comparison between the GFAM performance, and dFAM, FasART, and dFasART performances can be deduced from the summarized numbers of Table 4, with the cautionary comments about the fairness of this comparison that were mentioned above.

In all our experiments above, we used simulated or real databases that predominantly had input patterns of dimensionality 2. Furthermore, for the GFAM results reported (databases 1–19 of Table 1) we used these databases to identify good (default) GA parameter values. It is therefore worth reporting GFAM's performance on databases that have input patterns of higher than 2 dimensionality (see databases 20–27 of Table 1), and for which we use the good GA parameter values identified on databases 1–19 (see Table 1). The results (PCC on the test sets, and number of FAM categories created by GFAM) are depicted in Table 5 (in particular, Table 5 shows) the GFAM results for two α parameter values (α is the parameter that appears in the fitness function); an α value of 0.5 (this is the value used for the Table 3 results) and an α value of 0.1. Note that a smaller α parameter value allows for higher size GFAM networks that end up exhibiting higher accuracy (PCC) on unseen data.

In order for the reader to be able to evaluate how good the GFAM results are we refer the reader to the work by Lim, Loh and Shih (2000), where they compared the accuracy and size of a 33 classifiers belonging to the tree, statistical and neural types classifiers. Three of the datasets that Lim, Loh and Shih have experimented with are the Satellite, the

Table 4
Accuracy and size results achieved by GFAM and other ART networks

Database name	dFAM		FasART		dFasART		GFAM	
	PCC	Size	PCC	Size	PCC	Size	PCC	Size
4Ci/Sq	87.19	33.38	92.76	22.30	87.95	22.38	95.03	9
1Ci/Sq	88.90	12.34	96.30	63.42	92.78	30.50	97.77	7
1Ci/Sq/0.3:0.7	75.79	6.96	97.00	56.12	96.28	12.24	97.93	6
2Ci/Sq/20:30:50	95.23	12.96	96.27	57.68	94.91	24.32	97.57	5

Note: dFAM: Distributed Fuzzy ARTMAP, FasART, dFasART : Distributed FasART, GFAM : Genetic Fuzzy ARTMAP.

Table 5
Accuracy and size results achieved by GFAM on 8 UCI databases

	Database Name	GFAM (0.5)		GFAM (0.1)	
		PCC	Size	PCC	Size
20	OPTDIGITS	88.09	13	91.21	22
21	PENDIGITS	90.25	15	94.35	28
22	SAT	83.35	7	84.60	8
23	SEG	94.13	12	95.14	15
24	WAV	81.55	3	83.00	4
25	SHUTTLE	99.55	5	99.55	5
26	GLASS	75.00	9	67.19	9
27	PIMA	77.59	2	76.72	3

The results are recorded for 2 settings of the fitness function parameter: 0.5 and 0.1.

Segmentation and the Waveform datasets that GFAM has been tested on (see Table 5). The GFAM results on the Satellite dataset are: 83.35% (84.6%) classification accuracy, needing 7 (8) categories (see Table 5). The accuracy results reported on the Satellite dataset by Lim, Loh, and Shih are: Minimum classification accuracy of 60% and maximum classification accuracy of 90%. Furthermore the tree type classifiers (22 of them) created a minimum tree size of 8, while the median tree size was 63. Finally, the two most celebrated decision tree algorithms, CART and C4.5, created tree sizes of 63 and 216. The GFAM results on the Segmentation dataset are: 94.13% (95.14%) classification accuracy, needing 12 (15) categories (see Table 5). The accuracy results reported on the Segmentation dataset by Lim, Loh, and Shih are: Minimum classification accuracy of 48% and maximum classification accuracy of 98% (achieved by the nearest neighbor classifier, which performs no data compression). Furthermore the tree type classifiers (22 of them) created a minimum tree size of 6, while the median tree size was 39. Finally, the two most celebrated decision tree algorithms, CART and C4.5, created tree sizes of 69 and 42. The GFAM results on the Waveform dataset are: 81.55% (83%) classification accuracy, needing 3 (4) categories (see Table 5). The accuracy results reported on the Waveform dataset by Lim, Loh, and Shih are: Minimum classification accuracy of 52% and maximum classification accuracy of 85%. Furthermore the tree type classifiers (22 of them) created a minimum tree size of 3, while the median tree size was 16. Finally, the two most celebrated decision tree algorithms, CART and C4.5. created tree sizes of 14 and 54.

Overall, one can state that GFAM performs well on a number of classification problems achieving good classification accuracy at a network size that compares very favorably with a number of other ART-based and non-ART based classifiers.

6. Conclusions

In this paper, we have introduced yet another method of solving the category proliferation problem in ART. This method relies on evolving a population of trained Fuzzy ARTMAP (FAM) neural networks. We refer to the resulting architecture as GFAM.

We have experimented with a number of databases that helped us identify good default parameter settings for the evolution of FAM. We defined a fitness function that gave emphasis to the creation of a small size FAM networks which exhibited good generalization. In the evolution of FAM trained networks, we used a unique (and needed) operator; the delete category operator. This operator allowed us to evolve into FAM networks of smaller size. The network identified at the end of the evolutionary process (i.e., last generation) was the FAM network that attained the highest fitness value. Our method for creating GFAM resulted in a FAM network that performed well on a number of classification problems, and on a few of them it performed optimally.

GFAM was found to be superior to a number of other ART networks (ssFAM, ssEAM, ssGAM, safe micro-ARTMAP) that have been introduced into the literature to address the category proliferation problem in ART. More specifically, GFAM gave a better generalization performance (in almost all problems tested) and a smaller than or equal size network (in almost all problems), compared to these other ART networks, requiring reduced computational effort to achieve these advantages. In particular, in some instances the difference in classification performance of GFAM and these other ART networks was quite significant (as high as 10%). Furthermore, in some instances the ratio of the number of categories created by these other ART networks, compared to the categories created by GFAM, was large (as high as 5). Finally, some comparisons were also drawn between GFAM and dFAM, FasART, and dFasART, and other classifiers that led us to the conclusion that GFAM achieves good classification accuracy by creating an ART network whose size compares very favorably with the size of the other classifiers.

Obviously, the introduced method to evolve trained FAMs can be extended to other ART architectures, such as EAM and GAM, amongst others, without any significant changes in the approach followed.

Acknowledgments

This work was supported in part by the NSF grants: CRCD: 0203446, CCLI: 0341601, DUE: 05254209, IIS-REU: 0647120, and IIS-REU:0647018.

Appendix A. Terminology

- FAM: Fuzzy ARTMAP.
- EAM: Ellipsoidal ARTMAP.
- GAM: Gaussian ARTMAP
- ssFAM, ssEAM, ssGAM: Semi-suprvised Fuzzy ARTMAP, Semi-supervised Ellipsoidal ARTMAP, Semi-supervised Gaussian ARTMAP.
- M_a: The dimensionality of the input patterns in the training, validation and test sets provided to us by the classification problem under consideration.
- *Training set*: The collection of input/output pairs used in the training of FAMs that constitute the initial FAM population in GFAM.
- *Validation set*: The collection of input/output pairs used to validate the performance of the FAM networks during the evolution of FAMs from generation to generation.
- *Test set*: The collection of input/output pairs used to assess the performance of the chosen FAM network, after the evolution of FAMs is completed.
- PT: Number of points in the training set.
- PV: Number of points in the validation set.
- $\bar{\rho}_a^{\min}$: This is the lower limit of the baseline vigilance parameter used in the training of the FAM networks that comprise the initial population of the FAM networks.
- $\bar{\rho}_a^{\max}$: This is the upper limit of the baseline vigilance parameter used in the training of the FAM networks that comprise the initial population of the FAM networks.
- β_a: The choice parameter used in the training of the FAM networks that comprise the initial population of the FAM networks. This parameter is fixed, and chosen equal to 1.0.
- Pop_{size}: The number of chromosomes (FAM trained networks) in each generation.
- $\mathbf{w}_j^a(p) = (\mathbf{u}_j^a(p), (\mathbf{v}_j^a(p))^c)$: the weight vector corresponding to category j of the pth FAM network from the Pop_{size} trained FAM networks in a generation; $\mathbf{u}_j^a(p)$ corresponds to the lower endpoint of the hyperbox that the weight vector $\mathbf{w}_j^a(p)$ defines and $\mathbf{v}_j^a(p)$ corresponds to the upper endpoint of this hyperbox.
- $l_j(p)$: The label of category j of the p-th FAM network from the Pop_{size} trained FAM networks in a generation.
- PCC(p): The percentage of correct classification on the validation set exhibited by the pth FAM network from the Pop_{size} trained FAM networks in a generation.
- $N_a(p)$: The number of categories in the pth FAM network from the Pop_{size} trained FAM networks in a generation.
- $\text{Gen}_{\max}$: The maximum number of generations allowed for the FAM networks to evolve. When this maximum number is reached evolution stops and the FAM with the best fitness value on the validation set is reported.
- NC_{best}: Number of best chromosomes that the GFAM transfers from the old generation to the new generation (elitism).
- Cat_{del}: New genetic operator that deletes a category from a FAM chromosome.
- $P(\text{Cat}_{\text{del}})$, P(Mut): The probabilities of deleting and mutating a category.
- α: The parameter in the GA fitness function that defines the importance of the number of categories in a FAM network compared to the FAM's size; larger α values give more emphasis to the number of categories in a network compared to the network's generalization performance.
- Fit(p) : The fitness function of the pth FAM network.
- $\text{Cat}_{\min}$: The minimum number of allowed categories in a FAM network. It is set equal to the number of classes in the classification problem under consideration.
- PT: Number of points in the training set.
- PV: Number of data-points in the validation set.
- PTes: Number of points in the test set.
- PS: Number of network parameter settings to produce the best ART network (ART is ssFAM, ssEAM, ssGAM and safe micro-ARTMAP).

Appendix B. Complexity comparisons of GFAM and other ART networks

In this section we provide a fair comparison between the number of operations needed by GFAM and the number of operations needed by ssFAM. Similar considerations are valid when comparing the number of operations needed by GFAM versus the number of operations needed by ssEAM and ssGAM. The comparisons between GFAM and safe micro-ARTMAP are slightly different, and thus omitted, but some observations regarding these comparisons are made at the end of this appendix.

To begin let us remind ourselves that in both GFAM and ssFAM an element contributing to their computationally complexity is the training of a number of FAM networks. So, obviously an estimate of the computational complexity associated with the training of FAM is needed. Furthermore, an additional element contributing to the computational complexity of ssFAM is assessing the performance of the produced trained FAMs (corresponding to different values of FAM network parameter settings) to obtain the FAM that achieved the highest value of fitness. Finally, for GFAM an additional element contributing to its computational complexity is the evolution of the trained FAMs (for a number of generations) and their performance assessment in order to produce the FAM (at the last generation) that achieved the highest fitness value. In the following, we produce estimates for the computational complexity of each one of these elements. Throughout this paper we have assumed that the reader is familiar with the training phase of a FAM network, and this assumption is necessary here, as well, where the computational complexity calculation of a trained FAM is carried through.

Element 1: *Training of FAM networks (for ssFAM and GFAM)*

We assume that the reader is familiar of how the training of FAM works. During FAM's training for each one of the

training patterns in the training set (PT designates the number of training patterns) we have to compute the match function value of every category in the representation layer of FAM (N_a designates the number of categories in the representation layer of FAM). Then for the categories that pass the vigilance test (i.e., the value of their match function exceeds the value of the vigilance parameter) we have to compute the values of the choice function (at most N_a categories will pass the vigilance test). Eventually, once a category is found that passes the vigilance test and attains the maximum of the choice function values, its label is compared with the label of the input pattern presented to FAM. If the label matches, learning ensues, otherwise the process is repeated until we find a category in FAM that passes the vigilance, attains the maximum value of the choice function values and leads us to the correct label (the one that the input pattern should be mapped to). Note that in FAM, the number of categories, N_a created is a portion of the number of training patterns (designated as PT) presented to FAM. The process of presenting all the input patterns in the training set and proceeding, as described above, is referred to as one list presentation of FAM's training phase. So, it is not difficult to see that the computational complexity of one list presentation in FAM is equal to

$$O(N_a^2).$$

The entire training phase of FAM requires $O(N_a^2)$ computations for every list presentation. Hence, the computational complexity of FAM's training phase is equal to

$$O(N_a^2)$$

with the understanding that the constant of proportionality involved in the $O(N_a^2)$ expression is the product of the number of list presentations needed by FAM to converge to a solution, and the number representing the ratio of training patterns in the training set over number of categories created in the trained FAM (note that this number could be one or two or even more orders of magnitude large).

Element 2: *Testing of FAM network (ssFAM)*

In order to obtain the "best" ssFAM network (the performance of this network has been reported in Table 3), we have to train FAM for many parameter settings, and examine the fitness of the produced trained networks on an independent (rather than the training) set, referred to as the validation set. Assume, that the number of patterns in the validation set is equal to PV. Assume also that the number of parameter settings used to identify the best ssFAM is equal to PS.

In testing a single ssFAM network we have (for every pattern in the validation set) to go through the process of calculating the value of the match function attained by each category (node) in the trained FAM (this number was designated as N_a). For all those categories (nodes) that pass the vigilance test (i.e., the value of their match function exceeds the vigilance parameter) we also have to compute the value of the choice function, attained by the category. Hence, the testing of a single FAM network requires

$$\mathrm{PV} \cdot O(N_a)$$

calculations. To test PS of these trained FAM networks we obviously require

$$\mathrm{PS} \cdot \mathrm{PV} \cdot O(N_a)$$

calculations.

Concluding, we can state that the total number of calculations needed to produce the "best" ssFAM network is equal to

$$\mathrm{PS} \cdot O(N_a^2) + \mathrm{PS} \cdot \mathrm{PV} \cdot O(N_a). \tag{B.1}$$

Element 3: *Evolution of trained FAMs, testing of evolved FAMs (GFAM)*

In the evolution of trained FAMs we start with $\mathrm{Pop}_{\mathrm{size}}$ trained FAMs that we intend to evolve, and for every generation in the evolutionary process we evolve the chromosomes that represent these trained FAMs. The evolution of these trained FAMs involves (i) encoding the trained FAMs as chromosomes, (ii) applying a number of GA operators on the FAM-chromosomes, and (iii) decoding the FAM-chromosomes to FAMs. The computational complexity of this evolution from one generation of FAMs to the next generation of FAMs is equal to

$$O(N_a).$$

The computational complexity of testing these evolved FAMs in every generation is equal to

$$\mathrm{Pop}_{\mathrm{size}} \cdot \mathrm{PV} \cdot O(N_a)$$

where $\mathrm{Pop}_{\mathrm{size}}$ was defined earlier as the number of chromosomes in the GA population. Obviously, this process (evolution of FAMs, testing of evolved FAMs) needs to be repeated for as many times as the number of generations, which was denoted as$\mathrm{Gen}_{\max}$. Hence the computational complexity required for the evolution of FAMs to come up with best fitness FAM (in the last generation) is equal to:

$$\mathrm{Gen}_{\max} \cdot \mathrm{Pop}_{\mathrm{size}} \cdot \mathrm{PV} \cdot O(N_a).$$

Concluding, we can now state that the total number of calculations needed for the training, evolution and testing of FAMs in the GFAM approach is equal to

$$\mathrm{Pop}_{\mathrm{size}} \cdot O(N_a^2) + \mathrm{Gen}_{\max} \cdot \mathrm{Pop}_{\mathrm{size}} \cdot \mathrm{PV} \cdot O(N_a). \tag{B.2}$$

In comparing the computational complexities required to produce the best ssFAM network and the GFAM network (Eqs. (B.1) and (B.2)) we notice that:

- $\mathrm{PS} \gg \mathrm{Pop}_{\mathrm{size}}$, and
- $\mathrm{PS} > \mathrm{Gen}_{\max} \cdot \mathrm{Pop}_{\mathrm{size}}$.

As a reminder, in most of the experiments that we conducted with the other (than GFAM) ART networks PS $>$ 20,000. On the other hand, $\mathrm{Gen}_{\max} = 500$, $\mathrm{Pop}_{\mathrm{size}} = 20$. Hence, the above inequality statements are appropriately justified. The above two observations assure us that GFAM is more computationally efficient than the "best" ssFAM. Similar observations are valid if we compare the computational complexity of GFAM and the computational complexity associated with discovering the "best" ssEAM and ssGAM.

The computational complexity of the "best" safe micro-ARTMAP (whose results are reported in Table 3) is similar with the computational complexity of the "best" ssFAM, with one, worth mentioning, distinction. In the training phase of safe micro-ARTMAP the input patterns are presented to the ART architecture only in the first list presentation. In subsequent list presentations only a portion of these input patterns are presented to safe micro-ARTMAP. However, safe micro-ARTMAP requires some additional calculations during its training phase. So, for all practical purposes, we can still assume that the computational complexity of the training phase of safe micro-ARTMAP can be represented by the same formulas used to represent the computational complexity of the training phase of FAM. Obviously, the computational complexity of testing trained safe micro-ARTMAPs to discover the best safe micro-ARTMAP is given by the same formula used to discover the best trained FAM.

References

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

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* (pp. 2782–2787): *Vol. 4*.

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

Anagnostopoulos, G. C., & Georgiopoulos, M. (2002). Category regions as new geometrical concepts in Fuzzy ART and Fuzzy ARTMAP. *Neuarl Networks*, *15*(10), 1205–1221.

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*.

Bala, J., De Jong, K., Huang, J., Vafaie, H., & Wechsler, H. (1996). Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, *4*(3), 297–311.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Wadsworth.

Brotherton, T. W., & Simpson, P. K. (1995). Dynamic feature set training of neural nets for classification. In J. R. McDonnel, R. G. Reynolds, & D. B. Fogel (Eds.), *Evolutionary programming*: *Vol. IV* (pp. 83–94). Cambridge, MA: MIT Press.

Burton, A. R, & Vladimirova, T. (1997). Utilisation of an adaptive resonance theory neural network as a genetic algorithm fitness evaluator. In *Proceedings of the 1997 IEEE international symposium on information theory* (pages 209).

Cantu-Paz, E. (2002). Feature sub-set selection by estimation of distribution algorithms. In W. B. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishna, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, & N. Jonoska (Eds.), *GECCO 2002: Proceedings of the genetic and evolutionary computation conference* (pp. 302–310). San Francisco, CA: Morgan Kaufmann Publishers.

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

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

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.

Georgiopoulos, M., Huang, J., & Heileman, G. L. (1994). Properties of learning in ARTMAP. *Neural Networks*, *7*(3), 495–506.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

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).

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.

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

Hancock, P. J. B. (1992). Pruning neural networks by genetic algorithm. In I. Aleksander, & J. Taylor (Eds.), *Proceedings of the 1992 international conference on neural networks*: *Vol. 2* (pp. 991–994). Amsterdam, The Netherlands: Elsevier Science.

Hancock, P. J. B., Smith, L. S., & Phillips, W. A. (1991). A biologically supported errr-correcting learning rule. In T. Kohonen, K. Makisara, O. Simula, & J. Kangas (Eds.), *Proceedings international conference on artificial neural networks*: *Vol. 1* (pp. 531–536). Amsterdam, The Netherlands: North Holland.

Harp, S. A., Samad, T., & Guha, A. (1989). Towards the genetic synthesis of neural networks. In *Proceedings of the third international conference on genetic algorithms*. Morgan Kaufmann.

Inza, I., Larranaga, P., Etxeberria, R., & Sierra, B. (1999). Feature sub-set selection by Bayesian networks based optimization. *Artificial Intelligence*, *123*(1–2), 157–184.

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

Kelly, J. D. Jr., & Davis, L. (1991). Hybridizing the genetic algorithm and the K nearest neighbors classification algorithm. In *ICGA 1991* (pp. 377–383).

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

Lim, T., Loh, W., & Shih, Y. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, *40*, 203–229.

Liu, H., Liu, Y., Liu, J., Zhang, B., & Wu, G. (2003). Impulse force based ART network with GA optimization. In Neural networks and signal processing. *(Proceedings of the 2003 international conference on neural networks and signal processing*) (pp. 499–502): *Vol. 1*.

Marin, F. J., & Sandoval, F. (1993). Genetic synthesis of discrete-time recurrent neural networks. In *Lecture notes in computer science*: *Vol. 686*. *Proceedings international workshop artificial neural networks (IWANN 1993)*. Berlin, Germany: Springer-Verlag.

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

Miller, G. F., Todd, P. M., & Hedge, S. U. (1989). Designing neural networks using genetic algorithms. In: J. D. Scaffer (Ed.) *Proceedings of the third international conference on genetic algorithms.* San Mateo, CA: Morgan Kaufmann.

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

Palmes, P. P., Hayasaka, T., & Usui, S. (2005). Mutation-based genetic neural network. *IEEE Transactions on Neural Networks*, *16*(3).

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.

Punch, W. F., Goodman, E. D., ChiaShun, M. P. L., Hovland, P., & Enbody, R. (1993). Further research on feature selection and classification using genetic algorithms. In *International conference on genetic algorithms 93* (pp. 557–564).

Sexton, R. S., Dorsey, R. E., & Jonson, J. D. (1998). Toward global optimization of neural networks: A comparison of the genetic approach and back-propagation. *Decision Support Systems*, *22*(2), 171–185.

Siedlecki, W., & Sklansky, J. (1989). A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, *10*, 335–347.

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

Verzi, S. J., Heileman, G. L., Georgiopoulos, M., & 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).

Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing*, *14*, 347–361.

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

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

Yang, J., & Honavar, V. (1998). Feature subset selection using genetic algorithms. *IEEE Intelligent Systems*, *13*, 44–49.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423–1447.

Yao, X., & Liu, Y. (1998). Making use of a population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man and Cybernetics, B*, *28*, 417–425.