2006 IEEE International Conference on Fuzzy Systems
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

GFAM: A Genetic Algorithm Optimization of Fuzzy ARTMAP

A. Al-Daraiseh, M. Georgiopoulos, G. Anagnostopoulos, A. S. Wu, M. Mollaghasemi

Abstract— **Fuzzy ARTMAP (FAM) is currently considered to be one of the premier neural network architectures in solving classification problems. One of the limitations of Fuzzy ARTMAP that has been extensively reported in the literature is the category proliferation problem. That is Fuzzy ARTMAP has the tendency of increasing its network size, as it is confronted with more and more data, especially if the data is of noisy and/or overlapping nature. To remedy this problem a number of researchers have designed modifications to the training phase of Fuzzy ARTMAP that had the beneficial effect of reducing this phenomenon. In this paper we propose a new approach to handle the category proliferation problem in Fuzzy ARTMAP by evolving trained FAM architectures. We refer to the resulting FAM architectures as GFAM. We demonstrate through extensive experimentation that an evolved FAM (GFAM) exhibits good generalization, small size, and produces an optimal or a good sub-optimal network with a reasonable computational effort. Furthermore, comparisons of the GFAM with other approaches, proposed in the literature, that address the FAM category proliferation problem, illustrate that the GFAM has a number of advantages (i.e. produces smaller or equal size architectures, of better or as good generalization, with reduced computational complexity).**

I.INTRODUCTION

THE Adaptive Resonance Theory (ART) was developed by Grossberg (1976). One of the most celebrated ART architectures is Fuzzy ARTMAP (Carpenter et al, 1992), which has been successfully used in the literature for solving a variety of classification problems. One of the limitations of Fuzzy ARTMAP (FAM) that has been repeatedly reported in the literature is the category proliferation problem, which is tightly connected with the issue of overtraining.

Manuscript received January 31, 2006. This work was supported in part by the National Science Foundation (NSF) under grants CRCD 0203446 and CCLI 0341601.

A. Al-Daraiseh is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: creepymaster@yahoo.com).

M. Georgiopoulos is with the School of Electrical Engineering and Computer Science, Orlando, FL 32816, USA (phone: (407) 823-5338, fax: (407) 823 5835; e-mail: michaelg@mail.ucf.edu).

G. Anagnostopoulos is with the Department of Electrical and Computer Engineering, Florida Institute of Technology, Melbourne, FL 32901, USA (e-mail: georgio@fit.edu).

A. S. Wu is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: aswu@cs.ucf.edu).

M. Mollaghasemi is with the Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, FL 32816, USA (e-mail: mollagha@mail.ucf.edu).

A number of authors have tried to address the category proliferation/overtraining problem in Fuzzy ARTMAP. Amongst them we refer to the work by Verzi, et al., 2001, Anagnostopoulos, et al., 2003 and Gomez-Sanchez, et al., 2001, where different methods were introduced and evaluated, that allow Fuzzy ARTMAP categories to encode patterns that are not necessarily mapped to the same label.

In this paper, we propose the use of genetic algorithms (Goldberg, 1989) to solve the category proliferation problem in Fuzzy ARTMAP. 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, modify and exchange information with each other in order to perform certain tasks. Our approach starts with a population of trained FAMs. GA operators are then utilized to manipulate these trained FAM architectures in a way that encourages better generalization and smaller size architectures. The evolution of trained FAM architectures allows these architectures to exchange and modify their categories in a way that emphasizes smaller and more accurate FAM architectures. Eventually, this process leads us to a FAM architecture (referred to as *GFAM*) that has good generalization performance and creates networks of small size; all of these benefits come with the additional advantage of reasonable computational complexity.

Genetic algorithms have been extensively used to evolve artificial neural networks. For a thorough exposition of the available research literature in evolving neural networks the interested reader is advised to consult Yao, 1999. To the best of our knowledge there is no work conducted in the literature so far that has attempted to evolve FAM neural network structures, and that is the main focus of our effort.

The organization of this paper is as follows: In section 2 we present GFAM. In Section 3, we describe the experiments and the datasets used to assess the performance of GFAM, and we also compare GFAM to four other ART networks that attempted to resolve the category proliferation problem in Fuzzy ARTMAP. Finally, in Section 4, we summarize our work.

II. EVOLVING FAM NETWORKS (GFAM)

It is assumed throughout this paper that the reader is familiar with the Fuzzy ARTMAP (FAM) neural network architecture, its training phase, and its network parameters. We also assume that the reader is familiar with the geometrical interpretation of the weights in the FAM neural network (i.e., every category in FAM is represented by the lower and upper endpoints of a hyper-rectangle, that contains within its boundaries all the encoded patterns).

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. To evolve the initial population of the trained FAM networks GFAM utilizes tournament selection with elitism, as well as genetic operators such as crossover and mutation, and it introduces two special operators, named Cat_{add} and Cat_{del}. 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 assumption here is that we have available a training set that is used to train the FAM architectures, a validation set that is used to validate the performance of the FAM architectures evolved by the GA, and a test set that eventually assesses the performance of the best fitting FAM.

Step 1: The algorithm starts by initializing Pop_{size} FAM networks each one of them using a different value of the vigilance parameter $\bar{\rho}_a$. In particular, we first define $\bar{\rho}_a^{inc} = \dfrac{\bar{\rho}_a^{\max} - \bar{\rho}_a^{\min}}{Pop_{size} - 1}$, and then the baseline vigilance parameter of every network is determined by the equation, $\bar{\rho}_a^{\min} + i\,\bar{\rho}_a^{inc}$ where $i \in \{1, 2, ..., Pop_{size} - 1\}$. Meanwhile, GFAM allows the user to change the order of training pattern presentation automatically and randomly (as it is known, the order in which the training patterns are presented to a FAM network during its training phase affects the size and the performance of that network).

Step 2: We train Pop_{size} FAM networks with the baseline vigilance parameter values and order of training pattern presentation, as defined in Step 1. We assume that the reader is familiar of how training a FAM network is accomplished, and thus the details are omitted.

Step 3: 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 mix of real and integer numbers representation to encode the networks. Each FAM chromosome consists of two levels, level 1 containing all the categories of the FAM 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 Figure 1).

We denote the category of a trained FAM network by $\mathbf{w}_j^a(p)$, where $\mathbf{w}_j^a(p) = (\mathbf{u}_j^a(p), (\mathbf{v}_j^a(p)^c)$ and the label of this category by $l_j(p)$ for $1 \le j \le N_a(p)$ where the index p is such that $1 \le p \le Pop_{size}$,. In this step we also

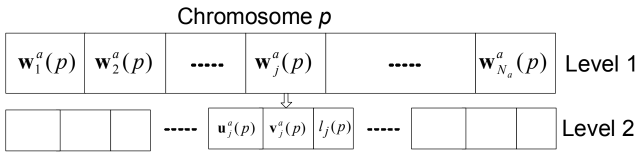


Fig. 1. GFAM Chromosome Structure

eliminate all the categories that encoded only one pattern in the training phase (single-point categories), referred to as cropping the chromosomes. Since our ultimate objective is to design a FAM network that reduces the network size and improves generalization we are discouraging at this stage the creation of single-point categories.

Step 4: In this step the GFAM evolves the chromosomes of the current generation.

Sub-step 4a: Calculate the fitness of each chromosome (trained FAM). This is accomplished by feeding into each trained FAM the validation set and by calculating the percentage of correct classification exhibited by each one of these trained FAM networks. In particular, if $PCC(p)$ designates the percentage of correct classification, exhibited by the p-th FAM, and this FAM network possesses $N_a(p)$ nodes in its category representation layer, then its fitness function value is defined by:

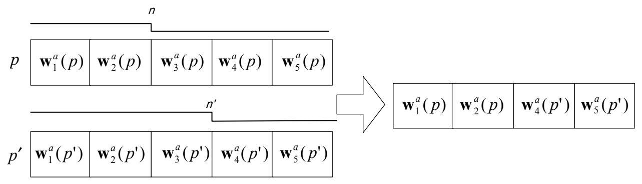
$$Fit(p) = \frac{(Cat_{\max} - N_a(p)) \cdot PCC^2(p)}{\dfrac{100}{Cat_{\min}} - \dfrac{PCC(p)}{N_a(p)} + \varepsilon}$$

The constant ε in the denominator of the above equation is a small positive constant and it is needed to make sure that the denominator would not be zero in the case when $N_a(p) = Cat_{\min}$ and $PCC(p) = 100$. In the above equation, $Cat_{\min}, Cat_{\max}$ are user defined minimum and maximum number of categories allowed to be created in an evolved FAM. This function was chosen as a fitness function after experimenting with other simple and complicated functions that did not perform as well. The chosen fitness function gave a good balance of optimizing both the size and the accuracy of the neural network.

Sub-step 4b: Initialize an empty generation (referred to as temporary generation).

Sub-step 4c: The algorithm searches for the best NC_{best} chromosomes from the current generation and copies them to the temporary generation.

Sub-step 4d: The remaining $Pop_{size} - NC_{best}$ chromosomes in the temporary generation are created by crossing over two parents from the current generation. The parents are chosen using a deterministic tournament selection method, 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 $\{1, 2, ..., N_a(p)\}$ and $\{1, 2, ..., N_a(p')\}$, respectively. Then, all the categories with index greater than index n' in chromosome with index p' and all the categories with index less than index n in the category 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 the following figure 2.



n
p
$\mathbf{w}_1^a(p)$ $\mathbf{w}_2^a(p)$ $\mathbf{w}_3^a(p)$ $\mathbf{w}_4^a(p)$ $\mathbf{w}_5^a(p)$
n'
p'
$\mathbf{w}_1^a(p')$ $\mathbf{w}_2^a(p')$ $\mathbf{w}_3^a(p')$ $\mathbf{w}_4^a(p')$ $\mathbf{w}_5^a(p')$
$\mathbf{w}_1^a(p)$ $\mathbf{w}_2^a(p)$ $\mathbf{w}_4^a(p')$ $\mathbf{w}_5^a(p')$

Fig. 2:. GFAM Crossover Implementation

Sub-step 4e: The operator Cat_{add} adds a new category to every chromosome created in step 4d with probability $P(Cat_{add})$. The new category has lower and upper endpoints **u**, **v** that are randomly generated as follows: For every dimension of the input feature space (M_a dimensions total) we generate two random numbers uniformly distributed in the interval [0, 1]; the smallest of the two random numbers is associated with the **u** coordinate along this dimension, while the largest of these numbers is associated with the **v** coordinate along this dimension. The label of this newly created category is chosen randomly amongst the N_b categories of the pattern classification task under consideration. A chromosome does not add a category if the addition of this category results in number of categories for this chromosome that exceeds the designated maximum number of categories $Cat_{\max}$.

Sub-step 4f: The operator Cat_{del} deletes one of the categories of every chromosome created in step 4e with probability $P(Cat_{del})$. A chromosome does not delete a category if the deletion of this category results in the number of categories for this chromosome to fall below the designated minimum number of categories, $Cat_{\min}$

Sub-Step 4g: In GFAM, every chromosome created by step 4f gets mutated as follows: with probability $P(mut)$ every category is mutated. If a category is chosen, its **u** or **v** endpoints 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 mean 0 and standard deviation 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 on level 2 of the chromosome structure, but the label of the chromosome is not mutated (the reason being that 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 5: If evolution has reached the maximum number $Gen_{\max}$ of iterations, then calculate the performance of the best-Fitness FAM network on the test set and report classification accuracy and number of categories that this Best-Fitness FAM network possesses. If the maximum number of iterations has not been reached yet, go to step 4 to evolve one more population of chromosomes.

III.GFAM EXPERIMENTS AND COMPARISONS WITH OTHER ART NETWORKS

Databases

To examine the performance of GFAM we performed a number of experiments on real and simulated datasets. Some of the specifics of these databases are given In Table 1, and more details about them are given below.

a) Gaussian Databases:

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

b) Structures within a Structure databases:

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 Parado-Hernandez, et al., 2003 paper where four different ART architectures were compared, Fuzzy ARTMAP, FasART, distributed Fuzzy ARTMAP, and distributed FasART. . In Table 1, *Ci* stands for circle and *Sq* stands for square, while *WN* means with noise and noise is taken to be at the level of 10%.

c) Real Databases:

These were obtained from the UCI repository (see Neuman, et al, 1998) and they are the well known IRIS, PAGE and ABALONE databases. It is worth mentioning that in the IRIS dataset we used only features 3 and 4, and we expanded its size (number of points) by simply adding noisy data into it.

TABLE I

Databases used in the GFAM experiments

	Database Name	# Numerical Attributes	# Classes (N_b)	% Major Class (A_0)
1	G2c-05	2	2	1/2
2	G2c-15	2	2	1/2
3	G2c-25	2	2	1/2
4	G2c-40	2	2	1/2
5	G4c-05	2	4	1/4
6	G4c-15	2	4	1/4
7	G4c-25	2	4	1/4
8	G4c-40	2	4	1/4
9	G6c-05	2	6	1/6
10	G6c-15	2	6	1/6
11	G6c-25	2	6	1/6
12	G6c-40	2	6	1/6
13	4Ci/Sq	2	5	0.2
14	4Sq/Sq	2	5	0.2
15	7Sq	2	7	1/7
16	1Ci/Sq	2	2	0.5
17	1Ci/Sq/0.3:0.7	2	2	0.7
18	5Ci/Sq	2	6	1/6
19	2Ci/Sq/5:25:70	2	3	0.7
20	2Ci/Sq/20:30:50	2	3	0.5
20	7SqWN	2	6	1/7
21	5Ci/SqWN	2	6	1//6
22	MOD-IRIS	2	2	1/2
23	ABALONE	7	3	1/3
24	PAGE	10	5	0.832

As we mentioned earlier, in all the experiments conducted with the aforementioned databases we had at our disposal a training set (used to design the trained ART network), a validation set (used to optimize the trained ART network), and a test set used to assess the performance of the optimized trained ART network.

Parameter Settings

We have experimented extensively with GFAM to identify a good set of parameters for the evolution of trained FAMs. We experimented with different numbers for the Pop_{size}, $Gen_{\max}$ and different values for the $P(Cat_{add})$, $P(Cat_{del})$, and $P(mut)$. The details of this experimentation are omitted due to lack of space. The GFAM results reported in this paper correspond to a GFAM 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). The FAM evolution used the following evolution parameters: $\rho_a^{\min}$ = 0.1, $\rho_a^{\max}$ = 0.95, β_a=0.1, Pop_{size} = 20, $Gen_{\max}$ = 500, NC_{best} = 3, $Cat_{\min}$ = 1, $Cat_{\max}$ = 300, $P(Cat_{add})$ =0.1, $P(Cat_{del})$=0.1, $P(mut)$ = $5/Na$.

Experimental Results

After running GFAM on the datasets included in Table 1, we identified the FAM network that attained the highest value of the fitness function at the last generation of the evolutionary process. Table 2 lists the accuracy of this GFAM network on the test set of the dataset under consideration. Table 2 also shows the size of this GFAM network. Furthermore, Table 2 reports the accuracy (on the test set) and the size of other ART architectures for the datasets included in Table 1.

TABLE II

Best Performance of All ART Algorithms (uAM: Safe uARTMAP; ssFAM: ss Fuzzy ARTMAP; ssEAM: ss Ellipsoidal ARTMAP; ssGAM: ss Gaussian ARTMAP; ss : semi-supervised version

	Database Name	GFAM		Safe uAM		ssFAM		ssEAM		ssGAM	
1	G2c-05	95.36	2	95.22	2	94.90	2	94.94	2	94.48	4
2	G2c-15	85.30	2	85.00	2	84.80	3	85.20	2	85.04	2
3	G2c-25	75.08	2	74.98	2	74.60	2	74.50	2	75.10	2
4	G2c-40	61.38	2	61.40	3	61.34	3	60.98	2	61.30	3
5	G4c-05	95.02	4	95.04	4	94.10	7	94.14	4	94.80	4
6	G4c-15	84.46	4	83.28	4	81.40	11	83.20	4	84.24	9
7	G4c-25	75.20	4	74.50	4	70.80	9	72.72	4	72.32	21
8	G4c-40	60.60	4	59.76	5	58.48	14	55.62	13	59.10	14
9	G6c-05	94.68	6	93.57	9	91.42	11	93.80	7	94.40	8
10	G6c-15	84.71	6	80.92	6	81.11	7	81.80	6	84.35	13
11	G6c-25	73.90	6	70.74	13	69.62	15	71.10	7	72.86	20
12	G6c-40	59.19	6	58.03	11	56.35	17	54.21	17	55.65	13
13	4Ci/Sq	96.32	8	95.42	8	87.23	18	94.68	5	93.4	12
14	4Sq/Sq	97.12	9	99.12	9	97.24	13	88.89	5	91.78	16
15	7Sq	97.2	7	97.22	16	97.26	16	88.5	19	95.83	93
16	1Ci/Sq	97.2	8	94.76	8	92.97	8	97.02	8	91.02	8
17	1Ci/Sq/ 0.3:0.7	97.8	8	96.82	8	93.21	8	97.13	8	92.33	8
18	5Ci/Sq	92	50	83.83	52	81.95	52	78.68	87	90.02	111
19	2Ci/Sq/ 20:30:50	97.87	3	97.22	6	90.24	12	97.01	3	95.6	9
20	7SqWN	87.3	7	86.67	20	80.15	24	75.23	32	83.11	123
21	5Ci/SqWN	81.97	50	71.72	52	68.39	57	69.2	136	81.3	145
22	MOD-IRIS	95.31	2	94.92	2	93.41	8	94.54	2	94.54	2
23	ABALONE	58.73	2	57.18	4	59.52	6	56.80	7	55.10	3
24	PAGE	95.59	3	88.82	6	90.63	3	89.54	3	89.34	5

In Table 2 above, we compare GFAM's performance with the performance of the following networks: ssFAM, ssEAM, ssGAM (see Anagnostopoulos, et al., 2003, Verzi, et al., 2001), and safe micro-ARTMAP (see Gomez, et al., 2002). We chose these networks for a reason. Each one of these ART networks at the time of their introduction into the literature emphasized that they were addressing the category proliferation problem in ART. More details about the specifics of each one of these networks can be found in their associated references. For the purposes of this paper it suffices to know that ssEAM covers the space of the input patterns with ellipsoids, while ssGAM covers the space of the input patterns with bell-shaped curves. Furthermore ssFAM, ssEAM, and ssGAM allow a category (hyper-rectangle or ellipsoid or hyper-dimensional bell shaped curve) to encode patterns of different labels provided that the plurality label of a category exceeds a certain, user-specified, threshold. Finally, safe micro-ARTMAP allows the encoding of patterns of different labels by a single category, provided that the entropy of the category does not exceed a certain, user-defined threshold.

In Table 2, the first column is the name of the database that we are experimenting with, while columns 2-6 of Table 2 contain the performance of the designated ART networks. The GFAM performance reported corresponds to the accuracy on the test set and the number of categories created by the FAM network that attained the highest value of the fitness function at the last generation of the evolutionary process. For the other ART networks the reported performance is the performance of the ART network that achieves the highest value of the fitness function amongst the trained ART networks with different network parameter settings (e.g., in ssFAM the best network was determined after training ssFAM networks with different values of the choice parameter, vigilance parameter, order of pattern presentation, and amount of mixture of labels allowed within a category).

According to the results in Table 2, in all instances (except minor exceptions) the accuracy of GFAM (generalization performance) is higher than the accuracy of the other ART network. According to the results in Table 2, in all instances (with no exceptions) the size of GFAM is smaller than the size of the other ART network (where ART is ssFAM, ssEAM, ssGAM or safe micro-ARTMAP), sometimes even by a factor of 15. For example, the generalization performance of GFAM can be as 13% better than the generalization performance of ssFAM, while its size can be by a factor of 4 times smaller than the size of ssFAM. Also, the generalization performance of GFAM can be as 13% better than the generalization performance of ssEAM, while its size can be by a factor of 4.5 times smaller than the size of ssEAM. Furthermore, the generalization performance of GFAM can be as 6% better than the generalization performance of ssGAM, while its size can be by a factor of 15 times smaller than the size of ssGAM. Finally, the generalization performance of GFAM can be as 10% better than the generalization performance of safe micro-ARTMAP, while its size can be by a factor of 3 times smaller than the size of safe micro-ARTMAP. Note that Figures 3a-3d also depict the comparisons of GFAM with other ART architectures in a pictorial fashion.

What is worth pointing out is that the better performance of GFAM is attained with reduced computations compared to the computations needed by the alternate methods (ssFAM, ssEAM, ssGAM, safe micro-ARTMAP). Specifically, the performance attained by GFAM requires training of 20 FAM networks, and evolving them for 500 generations (quite often evolving them for 500 generations is not needed). On the contrary, 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 20,000 experiments) and then choosing the network that achieved the higher value for the fitness function that we introduced earlier in the text. Of course, one can argue that such an extensive experimentation with these ART 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 ART network.

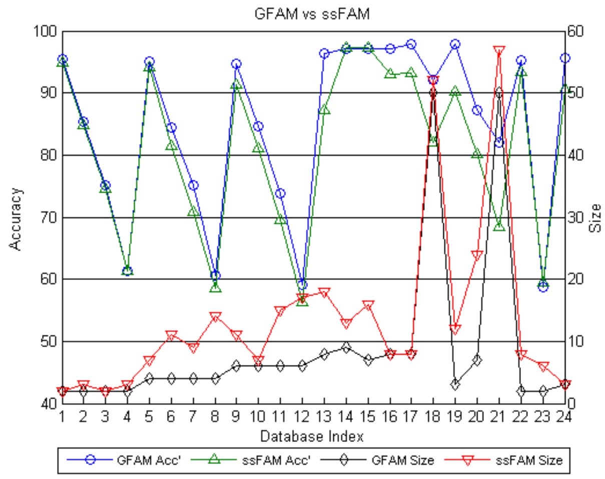


Fig. 3a. Accuracy and Size comparison of GFAM vs ssFAM

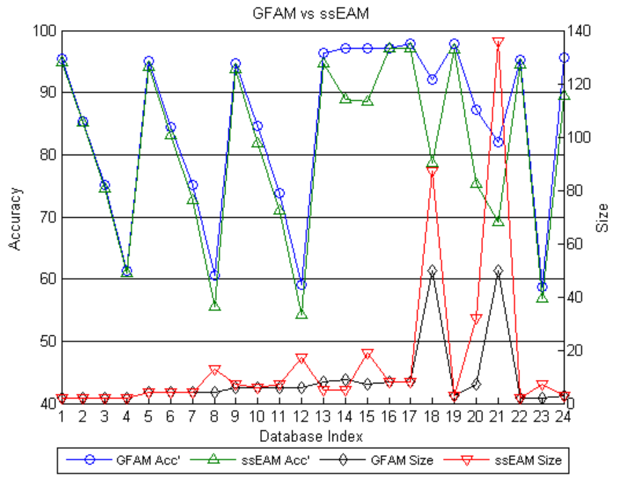


Fig. 3b. Accuracy and size comparison of GFAM vs ssEAM

IV. TIME COMPLEXITY ANALYSIS

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 section.

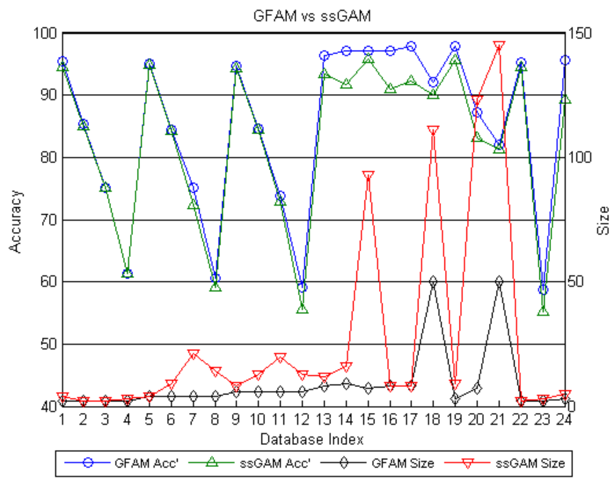


Fig. 3c. Accuracy and size comparison of GFAM vs ssGAM

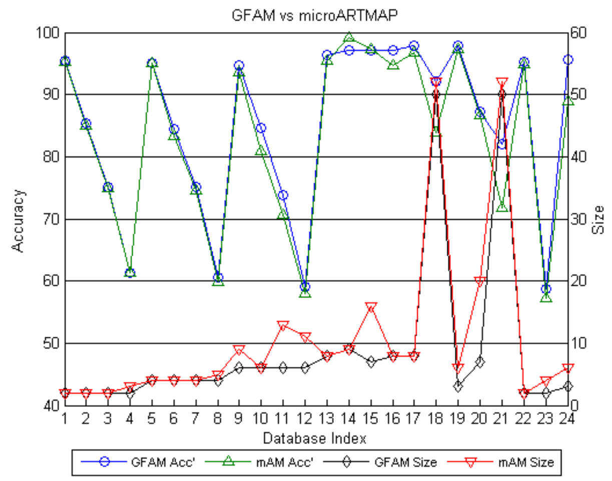


Fig. 3d. Performance and Size comparison of GFAM vs microARTMAP

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 are producing 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)

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 2), we have to train FAM for many parameter settings, and examine the fitness of the produced trained networks on an independent (than the training) set, referred to as 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

$$O(PV \cdot N_a)$$

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

$$O(PS \cdot PV \cdot N_a)$$

calculations.

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

$$O(PS \cdot N_a^2) + O(PS \cdot PV \cdot N_a)$$

Element 3: Evolution of trained FAMs, Testing of Evolved FAMs (GFAM)

In the evolution of trained FAMs we start with Pop_{size} trained FAMs. The computational complexity of this training is equal to

$$Pop_{size}\ O(N_a^2)$$

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

$$O(Pop_{size} \cdot PV \cdot N_a)$$

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 $Gen_{\max}$. Hence the computational complexity required for the evolution of FAMs to come up with best fitness FAM (in the last generation) is equal to:

$$O(Gen_{\max} \cdot Pop_{size} \cdot PV \cdot N_a)$$

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

$$O(Pop_{size} \cdot N_a^2) + O(Gen_{\max} \cdot Pop_{size} \cdot PV \cdot N_a)$$

In comparing the computational complexities required to produce the best ssFAM network and the GFAM network we notice that:

$$PS \gg Pop_{size}, \text{ and}$$

$$PS > Gen_{\max} \cdot Pop_{size}$$

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, $Gen_{\max} = 500, Pop_{size} = 20$. 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 2) 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.

In our experiments, while it took a specific computer to train and test a ssFAM network 20000 times around 6 to 18 *hours*, it took the same machine 3 to 30 *minuets* only to run GFAM on the same problem.

V. Conclusions

We introduced a new ART neural network architecture, named GFAM, produced by evolving a number of trained Fuzzy ARTMAP neural networks. The primary reason for introducing GFAM was to solve the category proliferation problem in Fuzzy ARTMAP.

We examined the performance of GFAM on a number of simulated and real datasets. The results illustrated that GFAM achieves good generalization (sometimes optimal generalization) while retaining a small network size. Comparisons of GFAM with other ART networks that addressed the category proliferation problem in Fuzzy ARTMAP revealed that GFAM achieves almost always better generalization and produces (all the time) a smaller (quite often significantly smaller) network size. The method used to create GFAM from trained ART networks can be extended to the evolution of other ART network architectures.

Appendix A – Terminology

- 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 (*PT* points).
- *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 (*PV* points).
- *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 (*PTes* points).
- $\overline{\rho}_a^{\min}$: 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.
- $\overline{\rho}_a^{\max}$: 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 0.1.
- Pop_{size} : The number of chromosomes (FAM trained networks) in each generation.
- $N_a(p)$: The number of categories in the p^{th} FAM network from the Pop_{size} trained FAM networks in a 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 p^{th} FAM network from the Pop_{size} trained FAM networks in a generation; $\mathbf{u}_j^a$ corresponds to the lower endpoint of the hyperbox that the weight vector $\mathbf{w}_j^a$ defines and $\mathbf{v}_j^a$ 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 p^{th} FAM network from the Pop_{size} trained FAM networks in a generation
- $Gen_{\max}$: 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 highest 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).
- $Cat_{\min}, Cat_{\max}$: The minimum and the maximum number of categories that a FAM chromosome is allowed to have during the evolutionary process that GFAM undergoes.
- Cat_{add}, Cat_{del} : New genetic operators that add and delete a category in a FAM chromosome.
- $P(Cat_{add}), P(Cat_{del})$, $P(Mut)$: The probabilities of adding, deleting and mutating a category in a FAM chromosome.
- 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).

ACKNOWLEDGMENT

This work was supported in part by a National Science Foundation (NSF) grant CRCD: 0203446. Georgios C. Anagnostopoulos and Michael Georgiopoulos also acknowledge the partial support from the NSF grant CCLI 0341601. Michael Georgiopoulos and Mansooreh Mollaghasemi also acknowledge the partial support from the NSF grant DUE: 0525429.

REFERENCES

[1] Anagnostopoulos, G.C., Bharadwaj, M., Georgiopoulos, M., Verzi, S.J., & Heileman, G. L. (2003). Exemplar-based pattern recognition via semi-supervised learning, in *Proc. of the International Joint Conference on Neural Networks*, Vol. 4, pp. 2782-2787, Portland, Oregon, July 20-24.

[2] 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 Trans. Neural Networks*, 3 (5), 698-713.

[3] Goldberg, D. E. (1989). Genetic Algorithms in search, optimization, and Machine Learning. Addison-Wesley, Reading, MA.

[4] 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, July 15-19.

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

[6] Neuman, 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.

[7] 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.

[8] Verzi, S.J., Georgiopoulos, M., Heileman, G.L., & Healy, M. (2001). Rademacher penalization applied to Fuzzy ARTMAP and Boosted ARTMAP, in *Proc. of the IEEE-INNS International Joint Conference on Neural Network*, pp. 1191-1196, Washington, DC, July 14-19.

[9] Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, Volume 87, Issue 9, Page(s):1423 - 1447