Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

Genetic Optimization of ART Neural Network Architectures

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

Abstract— **This paper focuses on the evolution of ARTMAP architectures, using genetic algorithms, with the objective of improving generalization performance and alleviating the ART category proliferation problem. We refer to the resulting architectures as GFAM, GEAM, and GGAM. We demonstrate through extensive experimentation that evolved ARTMAP architectures exhibit good generalization and are of small size, while consuming reasonable computational effort to produce an optimal or a sub-optimal network. Furthermore, we compare the performance of GFAM, GEAM and GGAM with other competitive ARTMAP architectures that have appeared in the literature and addressed the category proliferation problem in ART. This comparison indicates that GFAM, GEAM and GGAM have superior performance (generalize better, are of smaller size, and require less computations) compared with other competitive ARTMAP architectures.**

I. Introduction

Adaptive resonance theory was developed by Grossberg (see [9]). Some of the ART architectures that have appeared in the literature include Fuzzy ARTMAP (FAM) (see [4]), Ellipsoidal ARTMAP (EAM) (see [1]), and Gaussian ARTMAP (GAM) (see [11]). All of these ART architectures possess a number of desirable properties, such as they can solve arbitrarily complex classification problems, they converge quickly to a solution (within a few presentations of the list of input/output patterns belonging to the training set), they have the ability to recognize novelty in the input patterns presented to them, they can operate in an on-line fashion (new input patterns can be learned by the ART system without retraining with the old input/output patterns), and they produce answers that can be explained with relative ease.

One of the limitations of all the aforementioned ART architectures is the category proliferation problem. A number of researchers have addressed the category proliferation problem in ART, and amongst them we single out the work in [2], and [8], where different ways are introduced of allowing the ART categories to encode patterns that are not necessarily mapped to the same output (label). In this paper, we propose the use of genetic algorithms (see [7]) to solve the category proliferation problem in ART architectures, such as FAM, EAM and GAM. In our approach, we start from a collection of initially trained ART networks and evolve them using a fitness function that emphasizes good generalization and small network size. In an earlier work (see [6]) we evolved a collection of trained FAM networks; in this work we evolve FAM or EAM or GAM networks, we use a simpler fitness function and simpler GA operators than the ones used in [6], and we provide a good justification for the GA parameter values used.

The organization of the paper is as follows: In Section 2 we discuss ART in as much detail as it is needed for the reader to understand the evolution of ART architectures, which is the main theme of our effort. In Section 3, we explain the proposed approach to evolve ART architectures, from the genotype representation scheme, to the fitness function used, and from the special GA operators introduced, to the specific selection of appropriate GA parameter values. In Section 4, we discuss the experiments conducted and analyze the results produced. In Section 4, a thorough comparison of the genetically engineered ART structures and other ARTMAP architectures (see [2], [8]), which addressed the category proliferation problem, are included. This comparison indicates that GFAM, GEAM and GGAM compare favorably with these other ART architectures in the literature. Finally, in Section 5 we summarize our work and provide conclusive remarks.

II. ART Preliminaries

The Fuzzy ARTMAP (FAM) neural network architecture was introduced by Carpenter and Grossberg in their seminal paper [4]. Since its introduction, other ART architectures have been introduced into the literature. The focus in this paper is on Fuzzy ARTMAP and two other ART architectures: Ellipsoidal ARTMAP (see [1]) and Gaussian ARTMAP (see [11]). Our objective in this paper is to illustrate how we can design ART architectures from a population of FAMs, or EAMs, or GAMs. For simplicity we refer to all these ART architectures as ART and we use their specific name (FAM, or EAM or GAM) only when we want to discriminate one from the other. Similarly, we refer to the genetically optimized ART architectures as GART and we use their specific name (GFAM, GEAM and GGAM) only when we want to discriminate one from the other.

We assume that the reader is familiar with the FAM, EAM and GAM architectures. In this section we only provide the necessary information that is needed to understand the evolution of these ART structures, explained in detail in Section 3. For instance, it is worth mentioning that the weights (templates) in ART architectures represent compressed representations of the input patterns presented to the ART network during its training phase. These compressed representations have a geometrical interpretation. In particular, every node (category) in the category representation layer of Fuzzy ARTMAP (FAM) 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

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

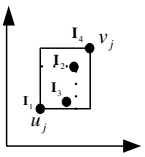

Fig. 1. A hyperbox category representation in FAM. This hyperbox has encoded patterns $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4$. In the figure, the $\mathbf{a}$ portion of these input patterns is depicted, as well as the lower end-point $\mathbf{u}_j$ and the upper endpoint $\mathbf{v}_j$ of this hyperbox.

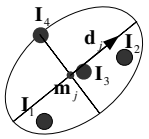

Fig. 2. An ellipsoidal category representation in EAM. This ellipsoid has encoded patterns $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4$. In the figure, the center point $\mathbf{m}_j$ and the direction vector $\mathbf{d}_j$ are shown, while the radius of the major axis, and the ratio of lengths of minor to major axis are easily deduced from the figure.

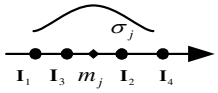

Fig. 3. A Gaussian curve category representation in GAM. This Gaussian distribution has encoded patterns $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4$. In the figure, the center point $\mathbf{m}_j$ and the standard deviation vector σ_j of the Gaussian curve are shown, while the number of points n_j that this Gaussian curve represents is easily deduced as being equal to 4.

FAM's training phase and were subsequently encoded by this category. In Figure 1, we show the hyperbox of a category in a FAM architecture (2-D example), with lower endpoint $\mathbf{u}_j$, and upper endpoint $\mathbf{v}_j$, and the input patterns (the $\mathbf{I}$'s that it has encoded). Also, every node (category) in the category representation layer of Ellipsoidal ARTMAP (EAM) has template weights that completely define an ellipsoid through its center, direction of major axis, length of the major axis, and ratio of lengths of minor axis to major axis in the ellipsoid. This ellipsoid includes within its boundaries all the input patterns that chose this category as their representative category in EAM's training phase and were subsequently encoded by this category. In Figure 2, we show the ellipsoid of a category in a EAM architecture (2-D example), with center $\mathbf{m}_j$, direction of the major axis $\mathbf{d}_j$, length of the major axis, represented by its radius r_j (implied from the figure), ratio of the lengths of minor axes to major axis μ (implied from the figure), and the input patterns $\mathbf{I}$'s that it has encoded. Finally, every node (category) in the category representation layer of Gaussian ARTMAP has template weights that define the mean vector, the standard deviation vector of a multi-dimensional Gaussian distribution, and the number of points that are associated with this Gaussian distribution. The mean vector of this Gaussian distribution and the standard deviation vector of this Gaussian distribution are defined in terms of the means and the standard deviations (across every dimension) of the points that chose this node (category) as their representative category, while the number of the points associated with this Gaussian distribution are the number of input patterns that chose this category as their representative category. In Figure 3, we show the Gaussian distribution of a category in a GAM architecture (1-D example), with mean $\mathbf{m}_j$, standard deviation σ_j, number of points n_j (in our example $n_j = 4$), and the input patterns (i.e., $\mathbf{I}$'s) that it has encoded.

It is also worth mentioning that the categories in FAM, EAM and GAM are allowed to expand up to a point allowed by a threshold, controlled by a network parameter denoted as the baseline vigilance parameter $\bar{\rho}_a$. This parameter assumes values in the interval [0, 1]. Small values of this parameter allow the creation of large categories, while large values of this parameter allow the creation of small categories. In the one extreme when $\bar{\rho}_a$ is equal to 0, a FAM or EAM category, 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. In GAM, small values of this parameter allow more and more patterns to be encoded by a GAM category, while large values of this parameter allow only a few patterns to be encoded by a GAM category. It turns out that this parameter has a significant effect on the number and type of categories formed, and consequently it affects the performance of these networks.

III. Evolution of ART Architectures

In the rest of the paper we assume that for every classification problem we are provided with a training set, a validation set, and a test set. The training set is used for the training of GFAM, GEAM, and GGAM architectures under consideration. The validation set is used to optimize the produced GFAM, GEAM or GGAM 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, GEAM, or GGAM network created.

GFAM, GEAM, and GGAM are evolved FAM, EAM, GAM networks, respectively, that are produced by applying, repeatedly, genetic operators on an initial population of trained FAM, EAM, or GAM networks. GFAM, GEAM, GGAM use tournament selection with elitism, as well as genetic operators, including crossover and mutation. In addition, GFAM, GEAM and GGAM use a special operator, Cat_{del}; this special operator is needed so that categories could be destroyed in the ART architectures, thus leading us, through evolution, to smaller ART structures.

In the process of discovering GFAM, GEAM or GGAM we are starting from a population of trained FAM, EAM or GAM networks that have been trained with different baseline vigilance parameter values, and different orders of pattern presentations of the training data (it has been a known fact that performance in ART is affected by the order according to which training data are presented to an ART architecture).

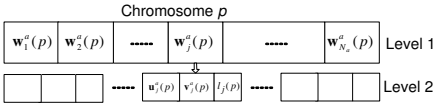

Fig. 4. GFAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information about the lower end-point, $\mathbf{u}_j^a$, and the upper end-point, $\mathbf{v}_j^a$, of the hyperbox corresponding to the category, as well as the label l_j of the category.

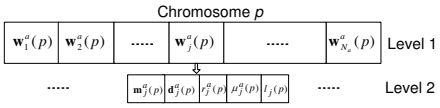

Fig. 5. GEAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center, $\mathbf{m}_j^a$, the direction vector of the major axis, $\mathbf{d}_j^a$, the radius (half length) of the major axis, r_j, and the ratio of the lengths of the minor axes over the length of the major axis, μ_j, of the ellipsoid corresponding to this category, as well as the label l_j of the category.

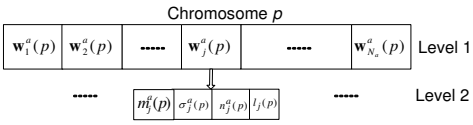

Fig. 6. GGAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center of the Gaussian curve, $\mathbf{m}_j^a$, the standard deviation vector of the Gaussian curve, σ_j^a, and the number of points represented by the Gaussian curve, n_j, as well as the label l_j of the category.

To better understand how ART (FAM, or EAM, or GAM) is genetically evolved, we resort to a step-by-step description of this process. The genetic evolution of ART can be articulated through a sequence of basic steps, defined succinctly below.

begin

1: Generate-Initial-Population()
2: Repeat
2.a. Evaluate-Fitness-And-Sort()
2.b. Selection()
2.c. CrossOver()
2.d. Catdel()
2.e. Mutate()
Until [max number of generations reached]

3. Return bestNetwork
end

Step 1: Generate Initial Population: The algorithm starts by training Pop_{size} ARTMAP networks (FAM, EAM or GAM), 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 $\bar{\rho}_a^{inc} = \frac{\bar{\rho}_a^{max} - \bar{\rho}_a^{min}}{Pop_{size}-1}$, and then the baseline vigilance parameter of every network is determined by the equation $\bar{\rho}_a^{min} + i\bar{\rho}_a^{inc}$, where $i \in \{1, 2, ..., Pop_{size} - 1\}$. The choice parameter in a FAM network was chosen to be equal to 0.1. The choice parameter in EAM network was chosen to be equal to 0.01. The initial value of the standard deviation γ in a GAM network is chosen to be equal to 0.6. In our experiments with GFAM and GEAM we chose $\bar{\rho}_a^{min} = 0.1$ and $\bar{\rho}_a^{max} = 0.95$, while in our experiments with GGAM we chose $\bar{\rho}_a^{min} = 0.1$ and $\bar{\rho}_a^{max} = 0.75$. 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, GEAM and GGAM use a real number representation to encode the networks. Each chromosome consists of two levels, level 1 containing all the categories of the network, and level 2 containing the template parameters needed to represent every category in level 1, as well as the label of every category in level 1. The chromosome encoding is explained in more detail in Figure 4 for GFAM, in Figure 5 for GEAM and in Figure 6 for GGAM.

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 $Fit(p)$, and it depends on the $PCC(p)$ and $N_a(p)$ values of this network. Note that, $PCC(p)$ 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 $Fit(p)$ of the p-th network is defined as follows:

$$Fit(p) = PCC(p) - 0.5(N_a(p) - Cat_{min}) \qquad (1)$$

Obviously, this fitness function increases as $PCC(p)$ increases or as $N_a(p)$ decreases. The value of Cat_{min} is chosen to be equal to the number of classes of the classification problem at hand. It is evident from the fitness equation that 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). This one of the simplest ways of defining a fitness function that depends on two measures (generalization of the network and size of the network) and has been extensively adopted in the classification literature (e.g., [3]).

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

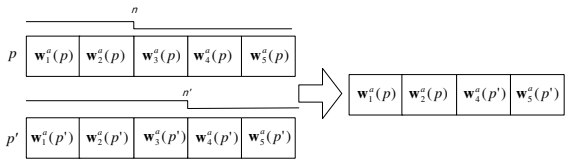

Fig. 7. GFAM, GEAM, GGAM Crossover implementation. In crossover the weight vectors of chromosome p, with index smaller than or equal to index n, and the weight vectors of chromosome p' with index larger than n', are combined (concatenated) to produce a new chromosome.

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. For each parent, p, p', a random cross-over point is chosen n, n'. Then, all the categories with index greater than n' in the chromosome p' and all the categories with index less than or equal to index n in the chromosome with index p are moved into an empty chromosome within the temporary generation. Notice that crossover is done at level 1 of the chromosome. This operation is pictorially illustrated in Figure 7.

Sub-step 2d (Category Deletion): The operator Cat_{del} deletes one of the categories of every chromosome (created in Step 2c) with probability $Pr(Cat_{del})$. 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 if the number of categories for this chromosome, due to deletion, falls below the designated minimum number of categories Cat_{min} the deletion is prohibited.

Sub-Step 2e (Category Mutation): Every chromosome created by step 2d gets mutated as follows:

In **GFAM**, with probability $Pr(Mut)$ every category is mutated. If a category is chosen to be mutated, either its $\mathbf{u}$ or $\mathbf{v}$ endpoint is selected randomly (with 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.

In **GEAM**, with probability $Pr(Mut)$ every category is mutated. If a category is chosen to be mutated, then every component of the ellipsoidal center $\mathbf{m}$ 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. Furthermore, the mutated category's axis ratio μ or radius r is selected with 50% probability. We add a small number, to the axis ratio or the radius, if they are chosen to be mutated. The small number is drawn from a Gaussian distribution with zero mean and standard deviation 0.01. However if μ, or r, due to mutation, become larger than 1, they are set back to the value of 1, while if they become smaller than zero we set their value to 0.0001.

In **GGAM**, with probability $Pr(Mut)$ every category is mutated. If a category is mutated, its mean vector $\mathbf{m}$, or standard deviation vector σ is selected randomly (50% probability). Then every component of this selected vector is 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 at level 2 of the chromosome structure. The label of the chromosome is not mutated because our initial GA population consists of trained networks, and consequently we have a lot of confidence in the labels of the categories that these trained networks have discovered through the training process.

Step 3: If evolution has not reached the maximum number of iterations, Gen_{max}, replace the current generation with the members of the temporary generation and go to step 2a. Otherwise calculate the performance of the best-fitness network on the test set.

IV. Experiments–Results

We conducted a number of experiments to assess the performance of the genetically engineered ART architectures. There were two objectives for this experimentation. The first one is to find proper values for the ranges of two of the GA parameters, the probability of deleting a category, $Pr(Cat_{del})$, and the probability of mutating a category, $Pr(mut)$. The second objective is to compare GFAM, GEAM and GGAM performance (for good parameter values) to that of popular ART architectures, such as ssFAM, ssEAM, ssGAM (see [2]) and micro-ARTMAP (see [8]).

A. Databases

We have experimented with 19 databases, 16 simulated databases and 3 real databases. Each dataset in the database was randomly divided into three subsets; training, validation and testing. The simulated databases include 12 Gaussian databases with 2-class, 4-class, and 6-class, and 5%, 15%, 25%, and 40% overlap between classes. The database denoted by 4Ci/Sq 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. The database denoted by 1Ci/Sq is the benchmark 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 database denoted by 30:70 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 database denoted by 20:30:50 is two circles in a square problem, a three class classification problem. One of the circles is smaller than the other. The probability

of finding a data point within the small circle, the large circle, and outside the circles but inside the square is 0.2, 0.3, and 0.5, respectively. The Modified Iris Database, the ABALONE database and the Pageblocks (PAGE) databases were obtained from the UCI repository (see [10]), and more details about these databases can be found there.

B. 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: the probability of deleting an ART category, $Pr(Cat_{del})$, and the probability of mutating an ART category, $Pr(Mut)$. A statistically sound approach was used to select good values of these GA parameters. The details are omitted due to lack of space. The rest of the GA parameters, such as Pop_{size}, Gen_{max}, and NC_{best} were chosen equal to 20, 500, and 3, respectively, after limited experimentation.

This analysis indicated that the best performing GA parameter setting for GFAM is $Pr(Cat_{del}) = 0.1$, and $Pr(Mut) = 0.4$, the best performing GA parameter setting for GEAM is $Pr(Cat_{del}) = 0.2$, and $Pr(Mut) = 0.4$, and the best performing GA parameter setting for GGAM is $Pr(Cat_{del}) = 0.4$, and $Pr(Mut) = 0.1$.

C. Comparison of GART with other ART Architectures

The second objective of our experimentation was to compare GFAM, GEAM, and GGAM 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, GEAM, GGAM with are: ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP. The first three are based on the principle of semi-supervision, discussed in [2]. Semi-supervision is a term attributed to learning in an ART architecture (FAM, EAM or GAM), 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 [8]) 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.

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 GART networks (see equation 1)). For the training of ssFAM, ssEAM, ssGAM, and safe micro-ARTMAP we used the same training set as the one used for the GART 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 GART 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 I.

The best parameter setting, identified in the previous subsection, for GFAM, GEAM, and GGAM 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, GEAM and GGAM for each database and its corresponding performance (PCC) on the test set are reported in Table I. The PCC results shown in Table I are truncated to one decimal place.

D. Observations from the Results

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

GFAM, GEAM and GGAM attain good performance on all the datasets, and quite often, optimal performance. The best performing network from the class of GART networks is GGAM. GGAM and GEAM outperform GFAM on all the structures, within structure problems. For all the other problems the differences between GEAM, and GGAM versus GFAM are not statistically significant. Furthermore, for most datasets at least one (if not all) the GART networks perform better (achieving higher PCC with fewer ART categories) compared to other ART architectures tested (ssFAM, ssEAM, ssGAM and safe micro-ARTMAP).

What is also worth pointing out is that the better performance of the GART 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, GEAM and GGAM cases we trained only a small number of these networks ($Pop_{size} = 20$ of them), and we evolved these trained ART networks for $Gen_{max} = 500$ generations, resulting in fewer computations than the ones needed for the ssFAM, ssEAM, ssGAM and safe micro-ARTMAP networks.

V. 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 ART networks, such as FAM, EAM or GAM. The evolution of trained ART networks creates an ART network, referred to as GFAM, or GEAM or GGAM.

TABLE I

BEST RESULTS OBTAINED FROM GFAM, GEAM AND GGAM COMPARED TO BEST RESULTS OBTAINED FROM SAFE μARTMAP, SSFAM, SSEAM AND SSGAM

Database	GFAM		GEAM		GGAM		Safe μAM		ssFAM		ssEAM		ssGAM	
Name	PCC	Cat	PCC	Cat	PCC	Cat	PCC	Cat	PCC	Cat	PCC	Cat	PCC	Cat
G2c-05	95.4	2	95.3	2	95.3	2	95.2	2	94.7	2	94.6	2	94.6	2
G2c-15	85.3	2	85.2	2	85.2	2	85.0	2	85.5	2	85.2	2	85.5	2
G2c-25	75.2	2	75.2	2	75.2	2	74.9	2	75.0	2	75.1	2	75.0	2
G2c-40	62.0	2	61.8	2	61.7	2	61.4	3	59.5	2	59.5	2	59.5	3
G4c-05	95.1	4	95.0	4	95.0	4	95.0	4	94.5	5	94.9	4	95.5	4
G4c-15	84.7	4	84.6	4	84.7	4	83.2	4	82.7	4	82.0	4	83.4	6
G4c-25	75.0	4	75.1	4	75.3	4	74.5	4	70.3	9	72.9	4	72.3	21
G4c-40	59.9	4	59.8	4	75.3	4	58.9	4	57.8	7	54.7	7	59.5	14
G6c-05	94.8	6	94.7	6	94.8	6	92.3	6	87.2	8	93.4	6	94.6	8
G6c-15	84.8	6	85.1	6	85.2	6	80.9	6	80.5	6	82.0	7	83.4	9
G6c-25	74.3	6	74.1	6	74.4	6	67.9	6	70.2	15	71.4	7	71.2	13
G6c-40	60.1	6	59.9	6	60.0	6	54.0	6	55.1	17	49.3	7	55.1	13
4Ci/Sq	95.0	9	99.1	7	98.9	6	95.4	8	87.2	18	94.6	5	93.4	12
1Ci/Sq	97.7	7	99.6	3	99.8	2	94.7	8	92.9	8	97.0	8	91.0	8
30:70	97.9	6	99.9	2	99.9	2	96.8	8	93.2	8	97.1	8	92.3	8
20:30:50	97.5	5	98.1	3	99.5	3	97.2	6	90.2	12	97.0	3	95.6	9
MOD-IRIS	95.3	2	95.3	2	94.9	2	94.9	2	94.7	8	94.7	2	94.7	2
ABALONE	61.8	3	62.2	3	62.6	3	58.1	3	60.0	6	58.8	3	56.3	2
PAGE	96.7	5	95.0	5	96.2	5	92.9	5	87.9	3	93.8	2	94.3	5

We have experimented with a number of databases that helped us identify good default parameter settings for the evolution of FAM, EAM or GAM. We defined a fitness function that gave emphasis to the creation of a small size ART networks which exhibited good generalization. In the evolution of ART trained networks, we used a unique (and needed) category operator, referred to as Cat_{del} operator (this operator allowed us to evolve into ART networks of smaller size). Our method for creating GFAM, GEAM and GGAM resulted in a networks that performed well on a number of classification problems, and on a few of them it performed optimally.

Furthermore, 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, GEAM, and GGAM gave a better generalization performance (in almost all problems tested) and a smaller than or equal size network (in all problems tested), compared to these other ART networks, requiring reduced computational effort to achieve these advantages. Finally, it is worth mentioning that the proposed approach of genetically engineering ART architectures can be applied to other exemplar-based neural network classifiers, such as Radial Basis Function (RBF) Neural Networks (see chapter 4 of [5]).

ACKNOWLEDGMENT

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

REFERENCES

[1] Anagnostopoulos GC (2001) Novel Approaches in Adaptive Resonance Theory for Machine Learning. Unpublished Doctoral Thesis, University of Central Florida, Orlando.

[2] G. C. Anagnostopoulos, M. Bharadwaj, M. Georgiopoulos, S. J. Verzi and G. L. Heileman, "Exemplar-based Pattern Recognition via Semi-Supervised Learning," Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '03), vol. 4, pp. 2782–2787, July 20–24, Portland, Oregon.

[3] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone. Classification and Regression Trees. Wadsworth, 1984.

[4] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architectur for Incremental Supervised Learning of Analog Multidimensional Maps," IEEE Transactions on Neural Networks vol. 3, pp. 698–713, 1992.

[5] C. Christodoulou and M. Georgiopoulos. Applications of Neural Networks in Electromagnetics. Artech House, January 2001.

[6] A. Al-Daraiseh, M. Georgiopoulos, A. S. Wu, G. Anagnostopoulos, and M. Mollaghasemi, "GFAM: A genetic optimization of Fuzzy ARTMAP", Proceedings of the 2006 WCCI Conference, Vancouver, Canada, July pp. 16–21.

[7] D. E. Goldberg, *Genetic Algorithms in search, optimization, and Machine Learning*, Addison-Wesley, Reading, MA

[8] E. Gomez-Sanchez, J. M. Cano-Izquierdo and J. Lopez-Coronado "Safe-uARTMAP: A new solution for reducing category proliferation in Fuzzy ARTMAP," Proceedings of the International Joint Conference on Neural Networks, 2001.

[9] S. Grossberg, "Adaptive Pattern Classification and Universal Recoding, II: Feedback, Expectation, Olfaction, and Illusions," Biological Cybernetics vol. 23, pp. 187–202, 1976.

[10] D. J. Newman, S. Hettich, C. L. Blake and C. J. Merz, *UCI Repository of machine learning databases* [http://www.ics.uci.edu/ mlearn/MLRepository.html] Irvine, CA: University of California, 1998.

[11] J. R. Williamson, "Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps," Neural Networks, vol. 9, pp. 881–897, 1996.