m-GFAM: An elegant approach to genetically optimize Fuzzy ARTMAP Neural Network Architectures

Assem Kaylani, Michael Georgiopoulos* and Mansooreh Mollaghasemi

University of Central Florida,
Orlando, Fl 32816, USA
**E-mail: michaelg@mail.ucf.edu*
www.ucf.edu

Georgios Anagnostopoulos

Florida Institute of Technology,
Melbourne, FL 32901, USA

Adaptive Resonance Theory (ART) neural network architectures, such as Fuzzy ARTMAP (FAM), have solved successfully a variety of classification problems. However, FAM suffers from an inherent problem that of creating larger architectures than it is necessary to solve the problem at hand (referred to as the *A*RT category proliferation problem). This problem is especially amplified for classification problems which have noisy data, and/or data, belonging to different labels, that significantly overlap. In this paper we introduce m-GFAM (modified genetically engineered Fuzzy ARTMAP), which is produced by evolving a population of FAM architectures. Our results demonstrate that m-GFAM successfully addresses the category proliferation problem by creating a small size trained ART structure that exhibits good generalization. Our experiments show that m-GFAM outperforms other ART architectures that have addressed the category proliferation problem before.

Keywords: Machine Learning, Classification, ARTMAP, Genetic Algorithms, Genetic Operators, Category Proliferation.

1. Introduction

The Fuzzy ARTMAP (FAM) neural network architecture was introduced by Carpenter and Grossberg in their seminal paper.[1] Since its introduction, other ART architectures have been introduced into the literature. 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, 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.

Learning in ART is accomplished by the creation of the hidden nodes (or units) referred to as categories. These categories represent compressed representations of the input patterns presented to the ART network during its training phase. Categories in FAM are hyperboxes with lower endpoint $\mathbf{u}_j$, and upper endpoint $\mathbf{v}_j$, which represent the minimum and maximum values of patterns that were encoded by this category. It is assumed that the reader is familiar with the FAM architecture, and most of the details are omitted due to lack of space.

A number of the proposed ART architectures suffer from the category proliferation problem, that is, the problem of creating an ART architecture that is larger than necessary to solve the problem at hand, especially when the data is noisy and/or of overlapping nature. In an earlier work,[2] genetic algorithms have been used to evolve a population of trained Fuzzy ARTMAP neural networks, creating an ART structure, referred to as GFAM. It was found that GFAM outperforms (in terms of generalization, size, and speed of producing the trained ART) a number of other ART architectures that have been previously proposed in the literature,[3],[4] addressing the same category proliferation problem. This effort expands and improves the work in[2] in many ways, such as: (a) a simpler, and easily understood fitness function is used to evolve a population of Fuzzy ARTMAPs, (b) the minimum possible number of GA operators is used, (c) the values of the GA parameters are, whenever possible, automatically and appropriately defined during the evolution, and (d) an automated stopping criterion is utilized to terminate the evolutionary process.

The organization of the paper is as follows: In section 2 we discuss how to evolve trained Fuzzy ARTMAPs to produce m-GFAM. In Section 3, we present experimental results, where we show that m-GFAM is producing as accurate and as small of a network as GFAM, but at reduced computational cost (a factor of 2 to 5 less computations are needed by m-GFAM). Finally, in Section 4 we summarize the work and we present conclusive remarks.

2. Evolution of Fuzzy ARTMAP

The evolution of FAM networks, is accomplished by applying, repeatedly, genetic operators on an initial population of trained FAM networks. The step-by-step description of this process is defined succinctly below.

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, and order of training pattern presentation (it has been a known fact that performance in ART is affected by the specific value of its baseline vigilance parameter, as well as the order of the training pattern presentation to the ART architecture). In our experiments with m-GFAM we chose the baseline vigilance range of $0.1 - 0.95$. The choice parameter was chosen to be equal to 0.1. Once the Pop_{size} networks are trained, they need to be converted to chromosomes so that they can be manipulated by the genetic operators. m-GFAM uses a real number, two-level representation to encode the networks, as explained in 1.

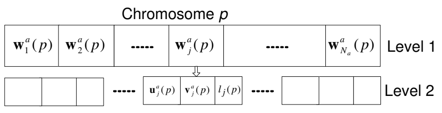

Fig. 1. GFAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information about the lower end-point, $\mathbf{u}_j^a$, and the upper end-point, $\mathbf{v}_j^a$, of the hyperbox corresponding to the category, as well as the label l_j of the category.

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 chromosome (ART trained network). In m-GFAM, we adopt a simplified fitness function compared to the one defined in GFAM. The fitness function for the p-th ART network is denoted by $Fit(p)$, and it depends on the percentage of correct classification, exhibited by the p-th network, on the validation set, $PCC(p)$, and the number of categories of the p-th network, $N_a(p)$. The fitness function is defined as follows (Cat_{min} is chosen to be equal to the number of classes of the classification problem at hand):

$$Fit(p) = PCC(p) - \alpha(N_a(p) - Cat_{min}) \tag{1}$$

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 (Prune): To be able to create smaller networks using the evolutionary search, we introduced a genetic operator, *Prune*, that deletes categories from a network using some selection criteria that tries to improve the efficiency of the genetic search by utilizing the knowledge we have about the performance of each category. For a network p, we delete a category j that is mapped to label k with probability $1 - CF_j^k(p)$, where $CF_j^k(p)$ is referred to as the confidence factory of this category. This confidence factor is defined as follows:

$$CF_j^k(p) = w_A A_j^k(p) + w_B S_j^k(p) \tag{2}$$

where, w_A and w_B were chosen to be equal to 0.5, Note that $A_j^k(p)$ is a measure of accuracy of classification achieved by category j, in the p-th network, that is mapped to label k. Furthermore, $S_j^k(p)$ is a measure of probability of selection of category j in the p-th network, that is mapped to label k. In particular, if the number of validation samples that selected this category, and were correctly classified by it, is denoted by $P_j^k(p)$, and the number of validation samples that selected this category is denoted by $C_j^k(p)$, then,

$$A_j^k(p) = \frac{P_j^k(p)/C_j^k(p)}{\max_j(P_j^k(p)/C_j^k(p))} \tag{3}$$

The probability of selection $S_j^k(p)$ of category j, of the p-th network, that is mapped to label k, is the number of validation patterns that selected this category, $C_j^k(p)$, divided by the maximum number of patterns $C_{j_{max}}^k(p)$ that selected any category j that predicts the same classification label, k, for the p-th network:

$$S_j^k(p) = C_j^k(p)/C_{j_{max}}^k(p) \tag{4}$$

Sub-Step 2d (Category Mutation): Every chromosome created by step 2c gets mutated as follows: For each category, either its **u** or **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, drawn from a Gaussian distribution. We use a Gaussian distribution that has mean of zero and a standard deviation that is equal to the severity factor that is calculated for every category based on its performance. The severity factor of a category depends on the network's temperature defined by the following expression:

$$T(p) = \frac{rank_{fitness}(p)}{Pop_{size}} \tag{5}$$

We use the following expression to control the severity of mutation:

$$SF_j^k(p) = \beta(1 - CF_j^k(p))T(p) \tag{6}$$

where β is a user defined parameter and was chosen to be equal to 1.0. The mutation technique described above eliminated the need for the user specified parameter $P(Mutate)$ defined in the original GFAM, which was problem dependent. In addition, the technique defined above has the added advantage of improving the efficiency of the genetic search.
Sub-step 2e (Cross-Over Operation): The remaining $Pop_{size} - NC_{best}$ chromosomes in the temporary generation are created by crossing over pairs of parents from step 2d. The parents are chosen using fitness-proportionate selection as follows: Assign the probability of selection for each chromosome to be proportional to its relative fitness. We randomly select two parents based on these probabilities. For each parent, p, p', a random cross-over point is chosen, designated as n, n', respectively. 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.

Our experimentation showed that fitness-proportionate selection yielded better results on most databases than those obtained using other selection methods such as random selection and deterministic tournament selection (which was used in the original implementation of GFAM).
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. One obvious stopping criterion is to set a threshold for the maximum number of generations, Gen_{max}, that the evolution is allowed to continue. 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 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 Gen_{max}. In our experiments we chose $Gen_{max} = 500$, and furthermore we stopped the evo-

lution if 50 generations (10% of Gen_{max}) elapsed without an appreciable network fitness improvement. Appreciable network fitness improvement is an improvement larger than 0.01.

3. Experiments and Results

We have experimented with 13 databases, 6 simulated databases and 7 real databases. Each dataset in the database was randomly divided into three subsets: training, validation and testing. The simulated databases include 2-D, Gaussian databases with 2-classes, 4-classes, and 6-classes, with 40% overlap between classes. The database denoted by 1Ci/Sq, 70:30 (1 circle in a square), 50:30:20 (2 circles in a square) are all motivated by the benchmark circle in the square problem, and the numbers in these designations refer to the probabilities of finding a data-point within a circle(s) or outside the circle(s) and inside the square. The remaining databases were obtained from the well known UCI repository,[5] and details about these databases can be found there.

For each of the 13 databases, we ran 10 replications of GFAM and m-GFAM, for 10 different seeds of the evolutionary process. The best results obtained from each algorithm are reported in Table 1. In the table, we have included the PCC (percentage of correct classification) and the number of categories of the *b*est fitting GFAM and m-GFAM network. We have also included in the table the time it took to run the 10 replications of GFAM and m-GFAM.

Database	**m-GFAM**			**GFAM**		
Name	*PCC*	*Cats*	*Time*	*PCC*	*Cats*	*Time*
G2c-40	61.4	2	80.8	60.9	2	143.4
G4c-40	59.7	4	135.9	59.5	4	229.1
G6c-40	59.8	6	242.8	59.9	6	337.7
1Ci/Sq	96.9	8	210.2	93.6	7	645.6
70:30	97.1	5	183.3	96.3	6	429.1
50:30:20	96.7	5	225.9	95.2	6	577.3
Iris	94.9	2	32.5	94.6	2	89.1
Abalone	61.7	3	69.6	62	3	126.7
PAGE	96.1	5	109.6	96	5	260.2
Opt	88.2	16	1185.9	88.6	11	3420.5
Sat	84.3	8	679.8	84	8	1434.1
Seg	93.0	15	90.0	94.7	13	428.8
Glass	75.0	9	3.8	71.9	8	20.3

4. Summary and Conclusion

In this paper, we have introduced an improved (compared to GFAM) genetically engineered Fuzzy ARTMAP neural network referred to as m-GFAM. Experimental results have shown that m-GFAM is as accurate and creates as small of an architecture as GFAM, while it does so at reduced computational cost (m-GFAM is 2 to 5 times faster than GFAM). Since, in an earlier paper[2] GFAM was shown to outperform other ART architectures[3,4] it turns out that m-GFAM outperforms these ART networks, as well. Furthermore, as we have explained in Section 2, m-GFAM finds good values for the needed genetic operators in an automated fashion, instead of an exhaustive experimentation (used in GFAM), and as a result it is a more elegant approach to evolve trained Fuzzy ARTMAPs. Finally, the proposed approach of genetically evolving Fuzzy ARTMAPs can be applied, with appropriate modifications, to other exemplar-based classifiers (i.e., classifiers that define, through some process, categories to compress the input data).

Acknowledgment

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

References

1. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds and D. B. Rosen, *IEEE Transactions on Neural Networks* **3**, 698 (1992).
2. A. Al-Daraiseh, M. Georgiopoulos, A. S. Wu, G. Anagnostopoulos and M. Mollaghasemi, GFAM: A genetic optimization of Fuzzy ARTMAP, in *Proceedings of the 2006 WCCI Conference*, (Vancouver, Canada, 2006).
3. G. C. Anagnostopoulos, M. Bharadwaj, M. Georgiopoulos, S. J. Verzi and G. L. Heileman, Exemplar-based pattern recognition via semi-supervised learning, in *Proceedings of the 2003 International Joint Conference on Neural Networks (IJCNN '03)*, (Portland, Oregon, USA, 2003).
4. E. Gomez-Sanchez, Y. Dimitriadis, J. Cano-Izquierdo and J. Lopez-Coronado, Safe-μARTMAP: A new solution for reducing category proliferation in Fuzzy ARTMAP, in *Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN '01)*, (Washington, DC, USA, 2001).
5. D. J. Newman, S. Hettich, C. L. Blake and C. J. Merz, UCI repository of machine learning databases (1998).