GFAM: Evolving Fuzzy ARTMAP Neural Networks

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

(*) University of Central Florida, Orl;ando, Fl
(**) Florida institute of Technology, Melbourne, FL
creepymaster@yahoo.com, michaelg@mail.ucf.edu, aswu@cs.ucf.edu, georgio@fit.edu, mollagha@mail.ucf.edu

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).

1. 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. A number of authors have tried to address the category proliferation/overtraining problem in Fuzzy ARTMAP. Amongst them we refer to the work the work by Verzi, et al., 2001, Anagnostopoulos, et al., 2003 and Gomez-Sanchez, et al., 2001, where different ways are introduced, and evaluated, of allowing the 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 (GA) to solve the category proliferation problem in Fuzzy ARTMAP. Genetic algorithms 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. In the Fuzzy ARTMAP setting we start with a population of trained FAMs. Then, GA operators are 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 at the expense 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 Xin, 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. In Section 4, we summarize our work.

2. Evolution of FAM Networks (GFAM)

It is assumed that the reader is familiar with the Fuzzy ARTMAP (FAM) neural network architecture, its training phase, and its network parameters. For every classification problem (dataset) that we experimented with we assume that we have a training set, a validation set and a test set.

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 typical genetic operations, such as tournament selection along 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, and explained in detail later.

Step 1: Initialize Pop_{size} number of FAM networks, each one of them operating with a different value for the vigilance parameter ρ_a, and possibly different order of pattern presentation.

Step 2: Train each one of the Pop_{size} initialized FAM networks, using the training set for a maximum number of iterations ($Gen_{\max}$).

Step 3: Convert the Pop_{size} trained FAM networks into chromosomes. Crop all chromosomes so that no one-point categories exist.

Step 4: Evolve the chromosomes of the current generation by executing the following sub-steps:

Sub-Step 4a: Calculate fitness for all chromosomes of the current generation.

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

Sub-Step 4c: Move the best (NC_{best}) chromosomes from the current generation to the temporary generation.

Sub-Step 4d: Select chromosomes for crossover from the current generation and thus further populate the temporary generation.

Sub-Step 4e: With a probability $P(Cat_{add})$ apply the Cat_{add} operator on every individual generated in sub-step 4d.

Sub-Step 4f: With a probability $P(Cat_{del})$ apply the Cat_{del} operator on every individual generated in sub-step 4e.

Sub-Step 4g: With a probability $P(Mut)$ apply the mutation operator on every individual generated in sub-step 4f.

Sub-Step 4h: Replace the current generation with the members of the temporary generation

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.

Each one of the aforementioned steps of the algorithm is now described in more detail, as needed.

Step 1 (More Details): The algorithm starts by training Pop_{size} FAM networks, each one of them trained with a different value of the vigilance parameter ρ_a. In particular, we first define $\rho_a^{inc} = \dfrac{\rho_a^{\max} - \rho_a^{\min}}{Pop_{size} - 1}$, and then the vigilance parameter of every network is determined by the equation $\rho_a^{\min} + i * \rho_a^{inc}$, where $i \in \{0, Pop_{size} - 1\}$. Meanwhile, GFAM allows the user to change the order of pattern presentation automatically and randomly.

Step 2 (More Details): We assume that the reader is familiar of how training a FAM networks is accomplished, and thus the details here are omitted.

Step 3 (More Details): 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).

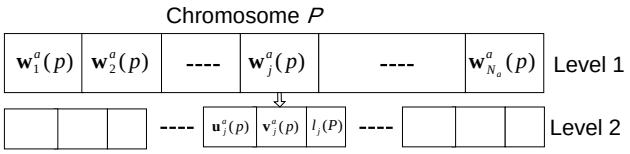

Figure 1: GFAM Chromosome Structure

We denote the category of a trained FAM network with index p $(1 \le p \le Pop_{size})$ 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 $1 \le j \le N_a(p)$. In this step we are also eliminating single-point categories in the trained FAM networks, 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 (More Details): In this step the GFAM applies a number of GA operators on the trained FAMs.

Sub-step 4a (More Details): 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:

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

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 $PCC(p) = 100.$

Sub-step 4b (More Details): Obvious, no further explanations are needed.

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

Sub-step 4d (More Details): 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 the 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.

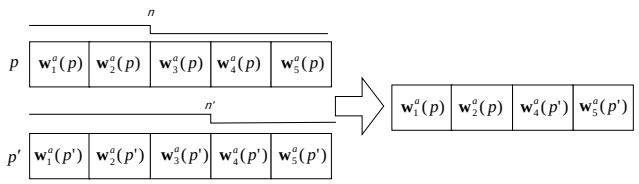

Figure 2: GFAM Crossover Implementation

Sub-step 4e (More Details): 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 $\mathbf{u}, \mathbf{v}$ 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 largest of these numbers is associated with the $\mathbf{v}$ coordinate along this dimension, while the smallest of the two random numbers is associated with the $\mathbf{u}$ 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 (More Details): 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 (More Details): 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 $\mathbf{u}$ or $\mathbf{v}$ 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 (More Details): Obvious, no more details are needed.

3. GFAM Experiments and Comparisons with other ART Networks

To examine the performance of GFAM we performed a number of experiments on real and simulated datasets. The

collections of simulated and real datasets are depicted in Table 1. The legend of Table 1 explains briefly the simulated datasets, while the real datasets were extracted from the UCI repository.

	Database Name	# Numerical Attributes	# Classes	% Major Class	Expected Accuracy
1	G2c-05	2	2	1/2	0.95
2	G2c-15	2	2	1/2	0.85
3	G2c-25	2	2	1/2	0.75
4	G2c-40	2	2	1/2	0.6
5	G4c-05	2	4	1/4	0.95
6	G4c-15	2	4	1/4	0.85
7	G4c-25	2	4	1/4	0.75
8	G4c-40	2	4	1/4	0.6
9	G6c-05	2	6	1/6	0.95
10	G6c-15	2	6	1/6	0.85
11	G6c-25	2	6	1/6	0.75
12	G6c-40	2	6	1/6	0.6
13	MOD-IRIS	2	2	1/2	0.95
14	ABALONE	7	3	1/3	0.6
15	PAGE	10	5	0.832	0.95
16	4Ci/Sq	2	5	0.2	1
17	4Sq/Sq	2	5	0.2	1
18	7Sq	2	7	1/7	1
19	1Ci/Sq	2	2	0.5	1
20	1Ci/Sq/0.3:0.7	2	2	0.7	1
20	5Ci/Sq	2	6	1/6	1
21	2Ci/Sq/5:25:70	2	3	0.7	1
22	2Ci/Sq/20:30:50	2	3	0.5	1
23	7SqWN	2	6	1/7	0.9
24	5Ci/SqWN	2	6	1//6	0.9

Table 1: Databases used in the GFAM experiments, where G*c_** represent a Gaussian dataset with * classes and ** overlap, 16-24 represent a shape within a shape dataset where Ci is a circle and Sq is a square; in the last two datasets WN means with noise (10%).

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.

3.1 Parameter Settings

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. 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: $\rho_a^{\min} = 0.1$, $\rho_a^{\max} = 0.95$, $\beta_a = 0.1$, $Pop_{size} = 20$, $Gen_{\max} = 500$, $NC_{best} = 3$, $Cat_{\min} = 1$, $Cat_{\max} = 300$, $P(Cat_{add}) = 0.1$, $P(Cat_{del}) = 0.1$, $P(mut) = 5/Na$.

3.2 Experimental Results

After running GFAM on the datasets in Table 1, we produce the accuracy and size of the GFAM network that attained the highest value of the fitness function at the last generation of the evolutionary process. Table 2 lists the accuracy and the size of this GFAM network as well as the accuracy and the size of other ART architectures for the same dataset.

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

Table 2: Best Performance of all ART Algorithms (uAM: Safe ARTMAP; ssFAM: ss Fuzzy ARTMAP; ssEAM: ss Ellipsoidal ARTMAP; ssGAM: ss Gaussian ARTMAP; ss : semi-supervised version

In Table 2 above we are comparing 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 (where ART is ssFAM, ssEAM, ssGAM or safe micro-ARTMAP). 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.

What is worth pointing out is that the better performance of GFAM is attained with reduced computation time compared to the computation time 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 network.

Furthermore, one might be confronted with a classification problem with a few data-points, where it might not be prudent to split the data-set into a training set and a validation set in order to come up with the optimum network parameters. 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.

4. Summary

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.

Acknowledgment

This work was supported in part by a National Science Foundation (NSF) grant CRCD: 0203446. Georgios

Anagnostopoulos and Michael Georgiopoulos acknowledge the partial support from the NSF grant CCLI 0341601.

References

Anagnostopoulos, G.C., Bharadwaj, M., Georgiopoulos, 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 Trans. Neural Networks*, 13 (1), 58-69.

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.

Gomez-Sanchez, E., Dimitriadis, Y.A., Cano-Izquierdo, J.M., Lopez-Coronado, J.: 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, 2001.

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

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.

Yao, X., "Evolving Artificial Neural Networks," Proceedings IEEE, Vol. 87, No. 9, pp. 1423-1447, 1997.

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).
- $\rho_a^{\min}$: This is the lower limit of the vigilance parameter used in the training of the FAM networks that comprise the initial population of the FAM networks.
- $\rho_a^{\max}$: This is the upper limit of the 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.
- $N_a(p)$: 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.