12

Genetically Engineered ART Architectures

Ahmad Al-Daraiseh[1], Assem Kaylani[1], Michael Georgiopoulos[1], Mansooreh Mollaghasemi[1], Annie S. Wu[1], and Georgios Anagnostopoulos[2]

[1] University of Central Florida, Orlando FL, 32816 michaelg@mail.ucf.edu
[2] Florida Institute of Technology, Melbourne, FL, 32796 georgio@fit.edu

Summary. This chapter 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 structures that have appeared in the literature and addressed the category proliferation problem in ART.

12.1 Introduction

Adaptive resonance theory (ART) was developed by Grossberg (see [18]). Some of the ART architectures that have appeared in the literature include Fuzzy ARTMAP (FAM) (see [10]), Ellipsoidal ARTMAP (EAM) (see [1]), and Gaussian ARTMAP (GAM) (see [36]). 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.

Since, Fuzzy ARTMAP's inception in 1992, a number of ART related papers have appeared in the neural network literature, some of which (as the ones mentioned above) modified the Fuzzy ARTMAP neural network so as to improve its performance. A related, important contribution in the ART literature is the one contributed by Petridis and Kaburlasos in 1998 ([31]), where they introduced the Fuzzy Lattice Neural Network (FLNN), a cross fertilization of fuzzy set theory and lattice theory, which can handle general

A. Al-Daraiseh: *Genetically Engineered ART Architectures*, Studies in Computational Intelligence (SCI) **67**, 233–262 (2007)
www.springerlink.com

types of data in addition to N-dimensional vectors. Most of the lattice work of Kaburlasos is included in a recently published book by Springer Verlag (see [21]) where a unified, cross-fertilizing approach for knowledge representation and modeling based on lattice theory is presented with emphasis on clustering, classification and regression applications. Some of Kaburlasos' recent work related with fuzzy lattice reasoning (FLR), which is the algorithm/software applied on a FLNN, can be found in [22].

The above references paint only an incomplete picture of the work that researchers have contributed into the ART literature, since Fuzzy ARTMAP's inception. However, since our goal in this chapter is to focus on the category proliferation problem in ART we are limiting, from this point on, our referrals to papers that have addressed this ART problem. Quite often the category proliferation problem in ART is connected with the issue of overtraining in ART. Over-training happens when ART is trying to learn the training data perfectly at the expense of degraded generalization performance (i.e., classification accuracy on unseen data) and also at the expense of creating many categories to represent the training data (leading to the category proliferation problem). Categories in ART are formed in order to compress the input data prior to mapping these compressed data to their respective outputs. The categories in Fuzzy ARTMAP are hyperboxes, in Ellipsoidal ARTMAP are ellipsoids, and in Gaussian ARTMAP they are Gaussian multi-dimensional probability distributions represented by their center points and widths (means and standard deviations) across every dimension. A number of authors have tried to address the category proliferation problem in ART. Amongst them we refer to the work by Marriott and Harrison, 1995, (see [28]), where the authors eliminate the match tracking mechanism of Fuzzy ARTMAP when dealing with noisy data; the work by Charalampidis et al., 2001 (see [13]), where the Fuzzy ARTMAP equations are appropriately modified to compensate for noisy data; the work by Verzi, et al., 2001 (see [35]), Anagnostopoulos, et al., 2002 and 2003 (see [2, 3]), and Gomez-Sanchez et al., (see [16, 17]), where different ways are introduced of allowing the ART categories to encode patterns that are not necessarily mapped to the same output (label); the work by Koufakou, et al., 2001, (see [25]), where cross-validation is employed to avoid the overtraining/category proliferation problem in Fuzzy ARTMAP; and the work by Carpenter, et al., 1998 (see [11]), Williamson, 1997 (see [37]), Parrado, et al., 2003 (see [30]), where the ART structure is changed from the winner-take-all to a distributed version and simultaneously slow learning is employed with the intent of creating fewer ART categories and reducing the effect of noisy patterns.

In this paper, we propose the use of genetic algorithms (see [15]) to solve the category proliferation problem in ART architectures, such as FAM, EAM and GAM.

Genetic algorithms (GAs) are a class of population-based stochastic search algorithms that are developed from ideas and principles of natural evolution. An important feature of these algorithms is their population based search

strategy. Individuals in a population compete and exchange information with each other in order to perform certain tasks. Genetic algorithms have been extensively used to evolve artificial neural networks. For a thorough exposition of the available research literature on evolving neural networks the interested reader is advised to consult the work of Yao (see [40]), 1999. In [40], the author distinguishes three different strategies in evolving neural networks. The first strategy is the one used to search for the weights of the neural network (see for example [32]). The second one is the one used to design the structure of the network (see for example [27] where only the structures are evolved, and [39], where both the structure and the interconnection weights are evolved), and the third one is the one where the learning rules of the neural network are evolved (see [19]). Furthermore, GAs may also be used to select the features that are input to the neural network. Since the pioneering work by Siedlecki and Sklanski (see [33]), genetic algorithms have been used for many selection problems using neural networks (see [6, 38]), and other classifiers, such as decision trees (see [4]), k-nearest neighbors (see [24]), and Naive Bayes classifiers (see [8, 20]).

To the best of our knowledge, work that utilizes GAs and ART neural network architectures is rather limited. For instance, in Liu, et al., 2003 (see [26]), a GA algorithm was employed to appropriately weigh attributes of input patterns before they were fed into the input layer of Fuzzy ARTMAP. The results reported reveal that this attribute weighting was beneficial because it produced a trained ART architecture of improved generalization. Furthermore, in Burton and Vladimirova, 1997 (see [7]), a Fuzzy ART neural network is employed as a GA fitness function evaluator, however the brevity of the published paper did not allow for the discussion of the details pertinent to this work.

In our work here, we use GAs to evolve simultaneously the weights, as well as the topology of the ART neural networks, such as FAM, EAM or GAM. In contrast to the feed-forward neural networks that have been extensively evolved, the aforementioned ART neural networks have a number of topological constraints, such as (a) they have one hidden layer of nodes, called category representation layer, and (b) every interconnection weight value from every node of the input layer to a node in the hidden layer is important (e.g., in FAM they are representing the minimum or the maximum of the values of input patterns across every dimension that were encoded by this node). Consequently, the only element of the ART's topology that can be evolved is the number of nodes in the category representation layer of the ART network. Furthermore, in our application we start with a population of trained ARTs, whose number of nodes in the hidden layer and the values of the interconnection weights converging to these nodes are fully determined (at the beginning of the evolution) by the ART's training rules. To this initial population of ART networks, GA operators are applied to modify these trained ART architectures (i.e., number of nodes in the hidden layer, and values of the interconnection weights) in a way that encourages better generalization and smaller size architectures.

It is worth reminding the reader that, as with many neural network architectures, the knowledge in ART is stored in its interconnection weights that have a very interesting geometrical interpretation (see [2, 9]. In particular, the interconnection weights in Fuzzy ARTMAP (converging to the nodes in the hidden layer) represent the lower and upper end-points of hyper-rectangles (referred to as categories) that enclose within their boundaries clusters of data that are mapped to the same label. Furthermore, the interconnection weights in Ellipsoidal ARTMAP represent the size and the direction of ellipsoids that enclose within their boundaries clusters of data that are mapped to the same label. Finally, the interconnection weights in Gaussian ARTMAP represent the mean vectors and variance vectors of Gaussianly distributed data that are mapped to the same label. Eventually, the evolution of these trained ARTs produces ART architectures, referred to as GFAM, GEAM or GGAM. GFAM, GEAM and GGAM are extracted from the last generation of the ART evolution, as the Fuzzy ARTMAP, Ellipsoidal ARTMAP, or Gaussian ARTMAP, respectively, that attained the highest fitness value. The fitness value of an ART network is defined in a way that attains higher values for better generalizing and smaller size ART networks.

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

The organization of this chapter is as follows: In section 12.2 we emphasize some of the basics of the ART architectures (FAM, EAM, GAM). In Sect. 12.3 we also provide the step-by-step description of the evolved ART architectures, such as GFAM, GEAM and GGAM, and we introduce the fitness function used for the evolution of ART architectures. In Sect. 12.4, we describe the experiments and the datasets used to assess the performance of GFAM, GEAM, and GGAM. In particular, in this section we explain the experiments that we conducted to come up with good default parameter values for the probability of category mutation, and for the probability with which a category is deleted in the evolution of ART architectures. Then for these good default parameter values we assess the performance of these genetically

engineered architectures, and we offer performance comparisons between the GFAM, GEAM and GGAM and other ART architectures that were proposed as solutions for the category proliferation problem in ART. In Sect. 12.5, we summarize our findings, and we offer some conclusive remarks.

12.2 ART Preliminaries

The Fuzzy ARTMAP (FAM) neural network architecture was introduced by Carpenter and Grossberg in their seminal paper (see [10]). Since its introduction, other ART architectures have been introduced into the literature and the focus on this paper is on Fuzzy ARTMAP and two other ART architectures Ellipsoidal ARTMAP (see [1]) and Gaussian ARTMAP (see [36]). Our objective in this paper is to illustrate how we can design genetically engineered ART architectures from a population of Fuzzy ARTMAPs, or Ellipsoidal ARTMAPs, or Gaussian ARTMAPs. We assume that the reader is familiar with all these ART architectures. In this section we will only describe the specifics of ART architectures that are needed to understand the genetically engineered ART structures, introduced in section 12.3. 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.

The block diagram of an ART architecture is shown in Fig. 12.1 (for FAM) and Fig. 12.2 (for EAM and GAM). Notice that this block diagram is simpler than the block diagram of FAM, reported in Carpenter and Grossberg in

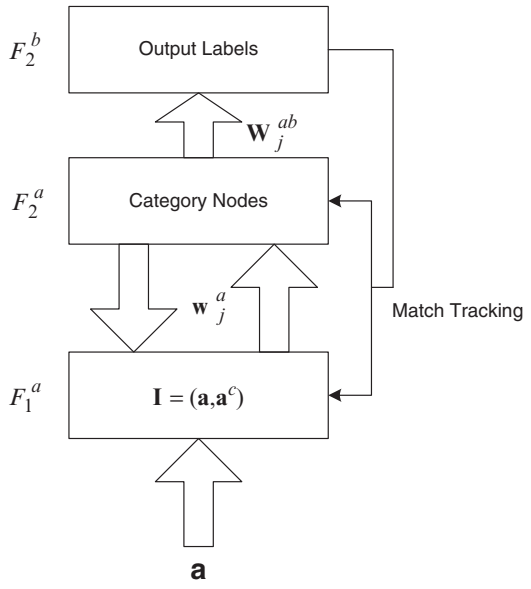

Fig. 12.1. The block diagram of a FAM Architecture

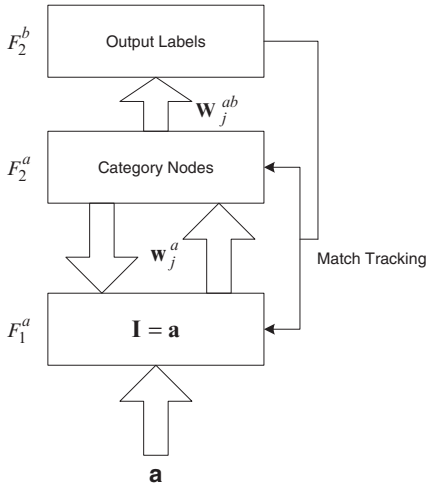

Fig. 12.2. The block diagram of an EAM or GAM Architecture

1992, and it has been adopted by various researchers in the field (e.g., Kasuba, 1993 [23], as well as Taghi, et al., 2003 [34]) because it can completely and succinctly describe the functionality of a variety of ART architectures, dealing with classification problems. As the focus of our paper is on classification problems, we also adopt this simpler ART architecture. The ART architecture, depicted in Fig. 12.1 (FAM) and Fig. 12.2 (EAM or GAM), has three major layers. The input layer F_1^a where the input patterns (designated by $\mathbf{I}$) are presented, the category representation layer F_2^a, where compressed representations of these input patterns (designated as $\mathbf{w}_j^a$), are formed, and the output layer F_2^b that holds the labels of the categories formed in the category representation layer. An additional layer, not shown in Fig. 12.1 and Fig. 12.2, and designated by F_0^a, is a pre-processing layer and its functionality is to pre-process the input patterns, prior to their presentation to ART. The first level of ART pre-processing takes the input vector and normalizes it so that each one of its components lies in the interval [0, 1], and that is the only level of pre-processing needed for EAM and GAM. The second level of pre-processing (needed only for FAM) takes the normalized input vector, referred to as $\mathbf{a}$ and complementary encodes it, by appending to it another vector, referred to as $\mathbf{a}^c$. The complement of vector $\mathbf{a}$ is defined as

$$\mathbf{a}^c = (1 - a(1), 1 - a(2), \ldots, 1 - a(M_a)) \tag{12.1}$$

where

$$\mathbf{a} = (a(1), a(2), \ldots, a(M_a)) \tag{12.2}$$

and M_a, in the above equations, stands for the dimensionality of the input pattern of the pattern classification task under consideration. It is worth

mentioning that the complementary encoding of the input patterns is necessary for the successful operation of Fuzzy ARTMAP (for an explanation see [14]), however it is not needed by either EAM or GAM. Therefore, in this paper we assume that the inputs to FAM are normalized and complementary encoded, while the inputs to EAM and GAM are simply normalized (see Fig. 12.1 for FAM, and Fig. 12.2 for EAM and GAM). Note that normalization of inputs prior to their presentation to a neural network is common practice in the neural network literature.

ART can operate in two distinct phases: the training phase and the performance (test) phase. The training phase of ART can be described as follows: Given a set of inputs and associated label pairs, $\mathbf{I}_1, label(\mathbf{I_1})$, $\mathbf{I}_2, label(\mathbf{I_2})$,..., $\mathbf{I}_{PT}, label(\mathbf{I_{PT}})$ (called the training set), we want to train ART to map every input pattern of the training set to its corresponding label. To achieve the aforementioned goal we present the training set to the ART architecture repeatedly. That is, we present $\mathbf{I}_1$ to F_1^a, $label(\mathbf{I}_1)$ to F_2^b, then $\mathbf{I}_2$ to F_1^a, $label(\mathbf{I}_2)$ to F_2^b, and finally, $\mathbf{I}_{PT}$ to F_1^a, $label(\mathbf{I}_{PT})$ to F_2^b. We present the training set to the ART network as many times as it is necessary for ART to correctly classify these input patterns. The task is considered accomplished (i.e., learning is complete) when the weights in ART do not change during a training set presentation, or after a specific number of list presentations is reached. The performance phase of ART works as follows: Given a set of input patterns (referred to as the test set), we want to find the ART output (label) produced when each one of the aforementioned test patterns is presented at its layer. In order to achieve this goal, we present the test set to the trained ART network and we observe the network's output. For the purposes of this paper, we assume that the reader knows the details of the training/performance phases in ART (FAM, EAM or GAM).

As we mentioned above, the weights (templates) in ART create 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 FAM's training phase and were subsequently encoded by this category. In Fig. 12.3, 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 axes 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 Fig. 12.4, we show the ellipsoid of a category in a EAM architecture

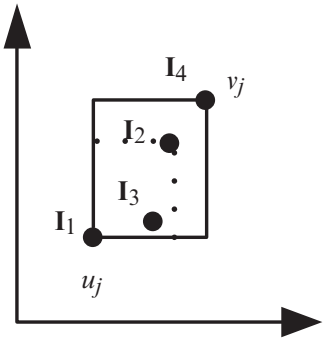

Fig. 12.3. A hyperbox category representation in FAM. This hyperbox has encoded patterns $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4$. In the figure, the **a** portion of these input patterns is depicted, as well as the lower end-point $\mathbf{u}_j$ and the upper endpoint $\mathbf{v}_j$ of this hyperbox

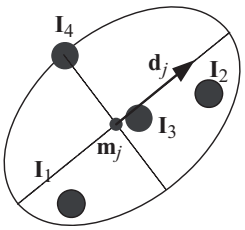

Fig. 12.4. An ellipsoidal category representation in EAM. This ellipsoid has encoded patterns $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4$. In the figure, the center point $\mathbf{m}_j$ and the direction vector $\mathbf{d}_j$ are shown, while the radius of the major axis, and the ratio of lengths of minor to major axis are easily deduced from the figure

(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 points that chose this category as their representative category. In Fig. 12.5, 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.

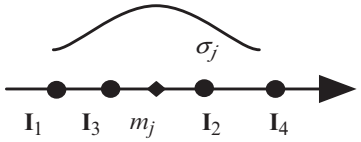

Fig. 12.5. A Gaussian curve category representation in GAM. This Gaussian distribution has encoded patterns $\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4$. In the figure, the center point $\mathbf{m}_j$ and the standard deviation vector σ_j of the Gaussian curve are shown, while the number of points n_j that this Gaussian curve represents is deduced as being equal to 4

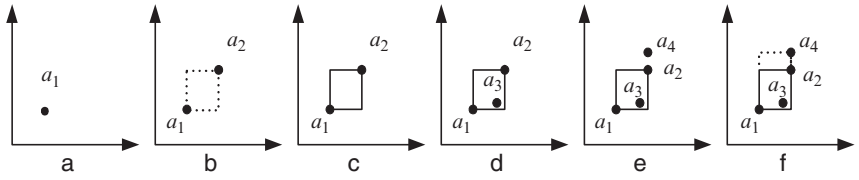

Fig. 12.6. FAM Learning (2-D Example): (a) A category with 0 size; (b) Introducing a new pattern $\mathbf{I}_2$, represented by $\mathbf{a}_2$; (c) The category expands to include $\mathbf{a}_2$; (d) Since new pattern $\mathbf{I}_3$, represented by $\mathbf{a}_3$ is inside the category, it doesn't change its size; (e) New Pattern $\mathbf{I}_4$, represented by $\mathbf{a}_4$ is presented; (f) Since $\mathbf{a}_4$ is outside the category, the category is expanded to include $\mathbf{a}_4$, within its boundaries

In essence, at the beginning of training, every category of FAM starts as a trivial hyperbox (equal to a point) and subsequently it expands to incorporate within its boundaries all the input patterns that in the training phase choose this hyperbox as their representative hyperbox, and are encoded by it (see Fig. 12.6, where the category expansion of FAM is shown for an example dataset). The size of hyperbox is the sum of the lengths of its sides.

Similarly, at the beginning of training, every EAM category starts as a trivial ellipsoid (equal to a point) and subsequently it expands to incorporate within its boundaries all the input patterns that in the training phase chose this ellipsoid as their representative ellipsoid, and are encoded by it (see Fig. 12.7, where the category expansion of EAM is shown for an example dataset). The size of the ellipsoid is measured as the length of the major axis.

Finally, at the beginning of training, every category of GAM starts as a collection of Gaussianly distributed data, with mean equal to the input pattern that was first encoded by this category, and a small standard deviation vector (part *a* of Fig. 12.8); as training progresses in GAM this GAM category is modified to incorporate the information of the additional input patterns that are encoded by it (see part *b* of Fig. 12.8 for an illustration of how the GAM category is modified for an example dataset). At any point in time the mean vector of this Gaussian distribution, corresponding to a category, is equal to the mean vector of all the input patterns encoded by the category, and the variance vector of the Gaussian distribution is equal to the variance vector corresponding to the input patterns that were encoded by the category.

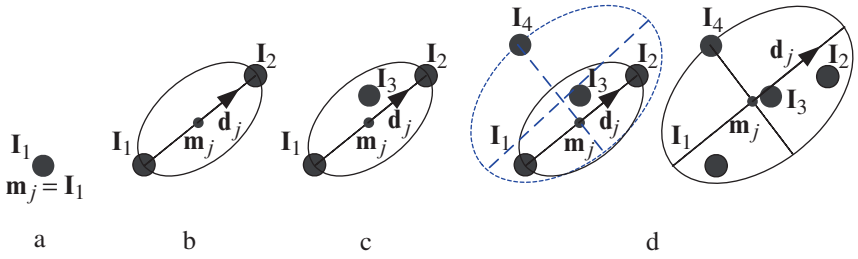

Fig. 12.7. EAM Learning (2-D Example): (a) A category with 0 size; (b) Introducing a new pattern $\mathbf{I}_2$; the category expands to include $\mathbf{I}_2$; (c) Introducing a new pattern $\mathbf{I}_3$; since the category includes $\mathbf{I}_3$, it does not change its size; (d) Pattern $\mathbf{I}_4$ is presented; since this pattern is outside the category, the category is expanded to include $\mathbf{I}_4$ within its boundaries

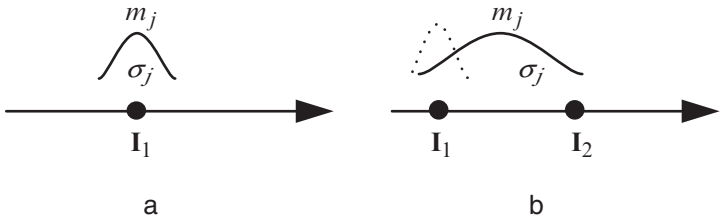

Fig. 12.8. GAM Learning (a) A category with 0 size; (b) Introducing a new pattern $\mathbf{I}_2$; the category characteristics (mean, standard deviation, of the Gaussian curve, as well as number of points encoded by the Gaussian curve) change to include the new knowledge that the new input pattern brings

The ability of the category to encode new input patterns depends on the Mahalanobis distance of an input pattern from the mean/variance vectors of the Gaussian distribution corresponding to the category.

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 . 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.

The performance of ART networks is measured in terms of the number of categories created in its training phase (small number of categories is good),

and how well it generalizes on unseen data (high generalization accuracy is good). In the process of discovering GFAM, 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).

The performance of Fuzzy ARTMAP (Ellipsoidal ARTMAP) is also affected by another network parameter called choice parameter. In the training of the Fuzzy ARTMAP (Ellipsoidal ARTMAP) networks used in our experiments, we fixed this choice parameter to the value of 0.1 (0.01). The performance of Ellipsoidal ARTMAP is also affected by another network parameter called length of minor to major axis parameter (denoted by the symbol μ), and expressing the ratio of lengths of minor axes versus major axis of the ellipsoid that corresponds to an EAM category. In the training of the Ellipsoidal ARTMAP networks used in our experiments, we fixed this choice parameter to the value of 1. The performance of Gaussian ARTMAP is also affected by another network parameter called initial variance parameter (denoted by the symbol γ), and representing the initial variance of a GAM category (after it has encoded a single input pattern). In the training of our Gaussian ARTMAP networks, we fixed this choice parameter to the value of 0.6.

12.3 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 test, 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.

To better understand how ART (FAM, or EAM, or GAM) is genetically designed, we resort to a step-by-step description of this design. The genetic design of ART can be articulated through a sequence of basic steps, defined succinctly below. Before we proceed, appropriate terminology is needed and is outlined in the Appendix to this chapter.

The following pseudo-code shows the basic steps of GFAM, GEAM and GGAM :

```
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 an EAM network was chosen to be equal to 0.01. The ratio of the length of the minor axes to major axes in EAM was chosen equal to 1. 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 $\bar{\rho}_a^{max} = 0.75$.

We assume that the reader is familiar with how training of FAM, EAM and GAM networks is accomplished, and thus the details here are omitted. Once the Pop_{size} networks are trained, they need to be converted to chromosomes so that they can be manipulated by the genetic operators. The following is a description of the encoding schemes used in GFAM, GEAM and GGAM:

- GFAM uses a real number representation to encode the networks. Each FAM chromosome consists of two levels, level 1 containing all the categories of the FAM network, and level 2 containing the lower and upper endpoints of the hyperboxes, representing every category in level 1, as well as the label of every category in level 1 (see Fig. 12.9). We denote a category of a trained FAM network with index p, $1 \leq p \leq Pop_{size}$, by $\mathbf{w}_j^p$, where $\mathbf{w}_j^a(p) = (\mathbf{u}_j^a(p), (\mathbf{v}_j^a(p))^c, l_j(p))$, where $\mathbf{u}_j^a(p)$, and $\mathbf{v}_j^a(p)$ are the lower and

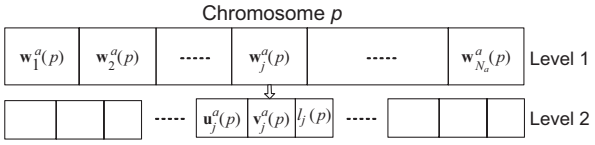

Fig. 12.9. GFAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information about the lower end-point, $\mathbf{u}_j^a$, and the upper end-point, $\mathbf{v}_j^a$, of the hyperbox corresponding to the category, as well as the label l_j of the category

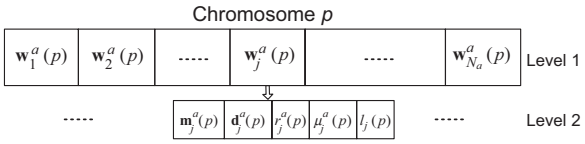

Fig. 12.10. GEAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center, $\mathbf{m}_j^a$, the direction vector of the major axis, $\mathbf{d}_j^a$, the radius (half length) of the major axis, r_j, and the ratio of the lengths of the minor axes 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

upper endpoints of the hyperbox corresponding to this category, and $l_j(p)$ is the label of this category.

- GEAM also uses a real number representation to encode the networks. Each EAM chromosome consists of two levels, level 1 containing all the categories of the EAM network, and level 2 containing the center, direction, radius of the major axis, the ratio of the minor axes to major axis of every category (ellipsoid) in level 1, as well as the label of the category (see Fig. 12.10). We denote the category of a trained EAM network with index p, $1 \leq p \leq Pop_{size}$, by $\mathbf{w}_j^a(p)$, where $\mathbf{w}_j^a(p) = (\mathbf{m}_j^a(p), \mathbf{d}_j^a(p), r_j^a(p), \mu_j^a(p), l_j(p))$. The first four components of this vector are the center, direction of the major axis, half length of the major axis, and ratio of the lengths of minor axes to major axis of the ellipsoid that represents this category, while the fifth component of this vector is the label of this category.
- GGAM also uses a real number representation to encode the networks. Each GAM chromosome consists of two levels, level 1 containing all the categories of the GAM network, and level 2 containing the mean, standard deviation, number of training patterns, and the label of every category (Gaussian curve) in level 1 (see Fig. 12.11). We denote the category of a trained GAM network with index p, $1 \leq p \leq Pop_{size}$, by $\mathbf{w}_j^a(p)$, where $\mathbf{w}_j^a(p) = (\mathbf{m}_j^a(p), \sigma_j^a(p), n_j^a(p), l_j(p))$. The first three components of this vector are the mean, standard deviation, and number of patterns that chose this category as their representative category, while the fourth component of this vector is the label of the category.

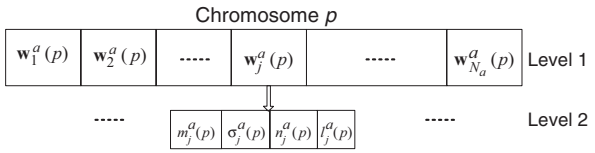

Fig. 12.11. GGAM chromosome structure. At level 2, the category's weight $\mathbf{w}_j^a$ contains the information of the center of the Gaussian curve, $\mathbf{m}_j^a$, the standard deviation vector of the Gaussian curve, σ_j^a, and the number of points represented by the Gaussian curve, n_j, as well as the label l_j of the category

In this step, we eliminate all single-point categories in the trained networks, referred to as cropping the chromosomes. Since our ultimate objective is to design a network that reduces the network size and improves generalization we discourage at this stage the creation of single-point categories. Our experiments have shown that cropping single-point categories is beneficial because it speeds-up the convergence of the GA.

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:

$$Fit(p) = PCC(p) - 0.5(N_a(p) - Cat_{min}) \tag{12.3}$$

  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 is 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., see [5]).
- **Sub-step 2b (Selection):** Initialize an empty generation referred to as the *temporary generation*. The algorithm searches for the best NC_{best} chromosomes (i.e., the chromosomes having the NC_{best} highest fitness values) from the current generation and copies them to the temporary generation without change.
- **Sub-step 2c (Cross-Over Operation):** The remaining $Pop_{size} - NC_{best}$ chromosomes in the temporary generation are created by crossing over

pairs of parents from the current generation. The parents are chosen using a deterministic tournament selection, as follows: Randomly select two groups of four chromosomes each from the current generation, and use as a parent, from each group, the chromosome with the best fitness value in the group. If it happens that from both groups the same chromosome is chosen then we choose from one of the groups the chromosome with the second best fitness value. If two parents with indices p, p' are crossed over, two random numbers n, n' are generated from the index sets $1, 2, ..., N_a(p)$ and $1, 2, ..., N_a(p')$, respectively. Then, all the categories with index greater than index n' in the chromosome with index p' and all the categories with index less than 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 Fig. 12.12.

- **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 **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. 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.

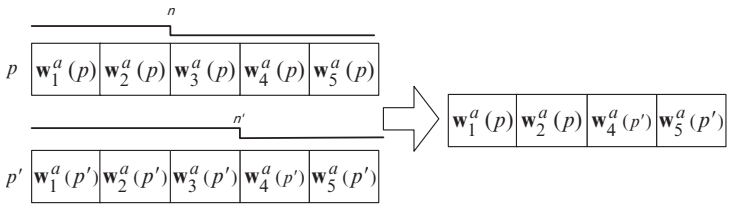

Fig. 12.12. GFAM, GEAM, GGAM Crossover implementation. In crossover the weight vectors of chromosome p, with index smaller than or equal to index n, and the weight vectors of chromosome p' with index larger than n', are combined (concatenated) to produce a new chromosome

- 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 **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 **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.

12.4 Experiments and 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, and micro-ARTMAP.

12.4.1 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. Each one of these datasets is described, in more detail, in the text that follows.

- **Gaussian Databases: (# 1-12)** 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 that 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 refer to these databases as **G#c-##** where the first number is the number of classes and the second number is the percentage of class overlap. For example, G2c-05 means that the Gaussian database is a 2-class and 5% overlap database.
- **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. Four 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 [30] where four different ART architectures were compared, Fuzzy ARTMAP, FasART, distributed Fuzzy ARTMAP, and distributed FasART.
  4Ci/Sq (# 13): This is a four circle in a square problem, a five class classification problem. The probability of finding a data point within a circle or inside the square and outside the circles is equal to 0.2. We refer to this database as **4Ci/Sq**.
  1Ci/Sq (# 14): This is a one circle in a square problem, a two class classification problem. The probability of finding a data point within a circle or inside the square and outside the circle is equal to 1/2. The sizes of the areas in the circle and outside the circle and inside the square are the same. This is the benchmark circle in the square problem. We refer to this database **1Ci/Sq**.
  1Ci/Sq/30:70 (# 15): This is a one circle in a square problem, a two class classification problem. The probability of finding a data point within a circle or inside the square and outside the circle is equal to 0.3 and 0.7, respectively. The sizes of the areas in the circle and outside the circle and inside the square are 0.3 and 0.7, respectively. This is a modified version of the circle in the square problem. We refer to this database as **30:70**.

2Ci/Sq/20:30:50 (# 16): This 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. We refer to this database as **20:30:50**.

- **Modified Iris (MOD-IRIS) Database (# 17):** In this database we started from the IRIS dataset (see [29]) of the 150 data-points, 3-class problem. We eliminated the data corresponding to the class that is linearly separable from the other 2 classes. Thus, we ended up with 100 data-points. From the four input attributes of this IRIS dataset we focused on only two attributes (attribute 3 and 4) because they seem to have enough discriminatory power to separate the 2-class data. Finally, in order to create a reasonable size dataset from these 100 points (so we can reliably perform cross-validation to identify the optimal ART, genetically engineered ART networks) we created noisy data around each one of these 100 data-points (the noise was Gaussian of zero mean and small variance) to end up with approximately 10,000 points. We named this database Modified Iris. We refer to this database as **MOD-IRIS**.
- **Modified Abalone (ABALONE) Database (# 18):** This database is originally used for prediction of the age of an abalone (see [29]). It contains 4177 instances, each with 7 numerical attributes, 1 categorical attribute, and 1 numerical target output (age). We discarded the categorical attribute in our experiments, and grouped the target output values into 3 classes: 8 and lower (class 1), 9-10 (class 2), 11 and greater (class 3). This grouping of output values has been reported in the literature before. We refer to this database as **ABALONE**.
- **Page Blocks (PAGE) Database (# 19):** This dataset represents the problem of classifying the blocks of the page layout in a document (see [29]). It contains 5473 examples coming from 54 distinct documents. Each example has 10 numerical attributes (e.g., height of the block, length of the block, eccentricity of the block, etc.,) and one target (output) attribute, representing the type of the block (text, horizontal line, graphic, vertical line, and picture). One of the noteworthy points about this database is that its major class (text) has a high probability of occurring (above 80%). This dataset has five classes, four of them make only 10% of the total instances. We refer to this database as **PAGE**.

The summarized specifics of each one of the aforementioned datasets are depicted in Table 12.1.

12.4.2 Selection of the GA parameters

As we have mentioned above, part the first objective of our experimentation was devoted to the selection of good values for the GA parameters: probability of deleting an ART category, $Pr(Cat_{del})$, and probability of mutating an

Table 12.1. Datasets used for experimentation, and their characteristics

Database	Name	# Training Instances	# Validation Instances	# Test Instances	# Attributes	# Classes	% Major Class
1	G2c-05	500	5000	5000	2	2	50.0
2	G2c-15	500	5000	5000	2	2	50.0
3	G2c-25	500	5000	5000	2	2	50.0
4	G2c-40	500	5000	5000	2	2	50.0
5	G4c-05	500	5000	5000	2	4	25.0
6	G4c-15	500	5000	5000	2	4	25.0
7	G4c-25	500	5000	5000	2	4	25.0
8	G4c-40	500	5000	5000	2	4	25.0
9	G6c-05	504	5004	5004	2	6	16.7
10	G6c-15	504	5004	5004	2	6	16.7
11	G6c-25	504	5004	5004	2	6	16.7
12	G6c-40	504	5004	5004	2	6	16.7
13	4Ci/Sq	2000	5000	3000	2	5	20.0
14	1Ci/Sq	2000	5000	3000	2	2	50.0
15	30:70	2000	5000	3000	2	2	70.0
16	20:30:50	2000	5000	3000	2	3	50.0
17	MOD-IRIS	500	4800	4800	2	2	50.0
18	ABALONE	501	1838	1838	7	3	33.3
19	PAGE	500	2486	2487	10	5	83.2

ART category, $Pr(Mut)$. As it is evident from our prior discussion there are a few other GA parameters that one has to carefully choose, such as Pop_{size}, Gen_{max}, and NC_{best}; we did not perform exhaustive experimentation to decide on the values of these parameters, but limited experimentation with these parameters for some of the above databases showed that reasonable choices for these parameters were: $Pop_{size} = 20$, $Gen_{max} = 500$, $NC_{best} = 3$.

Our approach to select good values for the GA parameters $Pr(Cat_{del})$, and $Pr(Mut)$ consisted of a number of steps delineated below:

- **Select GA Step 1:** We selected four different values for the $Pr(Cat_{del})$ to experiment with. These were: 0.05, 0.1, 0.2, and 0.4. We also selected four different values for the $Pr(Mut)$ to experiment with. These were: 0.0, 0.1, 0.2, and 0.4. This resulted in 16 combinations of parameter settings for $Pr(Cat_{del})$, and $Pr(Mut)$.
- **Select GA Step 2:** For each one of the 16 settings of the $Pr(Cat_{del})$, and $Pr(Mut)$ parameters, and for each of the 19 databases, described in an earlier section, we applied the GA optimization of FAMs, EAMs, and GAMs, as delineated in Sect. 12.2, 10 different times (using a different initial seed for the GA optimization). Consequently, for each database, and each parameter setting, and each of the genetically engineered ART algorithms we obtained 10 PCC and 10 N_a numbers.
- **Select GA Step 3:** For each genetically engineered ART algorithm (i.e., GFAM, GEAM, or GGAM), and each dataset, we chose the best-performing

(with respect to average validation PCC of the 10 experiments) parameter setting. Then, we used ANOVA statistical tests to choose other parameter settings that did not significantly differ (statistically) from the best performing parameter setting. We marked these parameter settings as good settings for this database and the associated GART (GFAM or GEAM or GGAM) algorithm.

- **Select GA Step 3:** After we performed Step 3 for all databases and all genetically engineered ART algorithms we counted the number of databases for each GART algorithm for which a particular parameter setting was deemed as "good" from the Select GA Step 3. Based on these counts we recommended the best parameter setting for each GART algorithm, and a range of acceptable parameter settings.

The following table (Table 12.2) summarizes the results for GFAM. Similar tables have been produced for GEAM and GGAM but are omitted due to lack of space. In Table 12.2 an entry of "1" for a database indicates that the corresponding parameter setting performed well (with respect to the average PCC on the validation set). An underscored "1" entry indicates that the corresponding parameter setting performed the best for this database (with respect to the average PCC on the validation set). In Table 12.2 the "1" entries corresponding to the Number of Categories criterion (actually average number of categories criterion) are omitted to preserve the table's clarity. However an entry of "1" for the PCC resulted also in an entry of "1" for the Number of Categories (not shown in Table 12.2). In Table 12.2 we designated with an asterisk the parameter setting that performed best for this database (with respect to the average Number of Categories criterion). A careful observation of the results shown in Table 12.2 indicate that any value of $Pr(Cat_{del})$ in the interval $[0.2, 0.4]$, and any value of the $Pr(Mut)$ in the interval $[0.05, 0.4]$ gives good results. Also, the results from Table 12.2 indicate that the best performing parameter setting for GFAM is $Pr(Cat_{del}) = 0.1$, and $Pr(Mut) = 0.4$, since for this parameter setting we observe the highest number of good performances (19), and best performances (7) of the associated GFAM (the count of the best performances consider the best observed performances with respect to the average PCC on the validation set or the average number of categories). Finally, we can also deduce from the results of Table 12.2 that a probability of mutation equal to 0 is not recommended, since it always (for all databases) results in a GFAM network that is not performing well.

From similar tables produced for GEAM and GGAM (omitted due to lack of space) we can draw similar conclusions. In particular, a careful observation of the GEAM results indicate that any value of $Pr(Cat_{del})$ in the interval $[0.2, 0.4]$, and any value of the $Pr(Mut)$ in the interval $[0.05, 0.4]$ gives good results for GEAM. Also, the best performing parameter setting for GEAM is $Pr(Cat_{del}) = 0.2$, and $Pr(Mut) = 0.4$, since for this parameter setting we observe the highest number of good performances (19), and best performances (6) of the associated GEAM. Finally, a probability of mutation equal to 0

Table 12.2. Goodness of Parameter Settings for the GA parameters $Pr(Cat_{del})$, and $Pr(Mut)$. An entry of "1" in the table indicates that the corresponding parameter setting is good, while the lack of an entry "1" indicates that the parameter setting is not good. The column before last counts the number of "1" entries for a particular parameter setting that are good, while the last column counts the number of times that a particular parameter setting is the best of all parameter settings (with respect to the average PCC on the validation set or the average number of categories). Entries are depicted for the PCC value and not for the number of categories value, in order to avoid cluttering the table; however whenever there is a "1" or lack of a "1" entry for the PCC value there is also a "1" or lack of a "1" entry for the number of categories value. Underscored "1" entries in this table designate parameter settings for which we obtained the best result related to the average PCC value, while asterisks designate parameter settings for which we obtained the best result with respect to the average number of categories value

Description	Data	g2c-05	g2c-15	g2c-25	g2c-40	g4c-05	g4c-15	g4c-25	g4c-40	g6c-05	g6c-15	g6c-25	g6c-40	4cirinsq	cirInSq	cins-70-30	2cins-50-30-20	Iris	abalone	pageblocks	sum	best
$Pr(Cat_{del}) = 0.05$	PCC	1																			1	
$Pr(Mut) = 0.0$	Num Cats																					
$Pr(Cat_{del}) = 0.05$	PCC	1	1	1	1	1	1	1	1	1	1	1		1		1	1	1	1	1	17	4
$Pr(Mut) = 0.1$	Num Cats																					
$Pr(Cat_{del}) = 0.05$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	
$Pr(Mut) = 0.2$	Num Cats																			*		1
$Pr(Cat_{del}) = 0.05$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	2
$Pr(Mut) = 0.4$	Num Cats																					
$Pr(Cat_{del}) = 0.1$	PCC																				0	
$Pr(Mut) = 0.0$	Num Cats																					
$Pr(Cat_{del}) = 0.1$	PCC	1	1	1	1	1	1	1		1	1	1		1		1	1	1	1	1	16	
$Pr(Mut) = 0.1$	Num Cats																					
$Pr(Cat_{del}) = 0.1$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1
$Pr(Mut) = 0.2$	Num Cats																					
$Pr(Cat_{del}) = 0.1$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	
$Pr(Mut) = 0.4$	Num Cats																	*		*		7
$Pr(Cat_{del}) = 0.2$	PCC	1																			1	
$Pr(Mut) = 0.0$	Num Cats																					
$Pr(Cat_{del}) = 0.2$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1		1	17	1
$Pr(Mut) = 0.1$	Num Cats																					
$Pr(Cat_{del}) = 0.2$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	
$Pr(Mut) = 0.2$	Num Cats																					1
$Pr(Cat_{del}) = 0.2$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1
$Pr(Mut) = 0.4$	Num Cats																					
$Pr(Cat_{del}) = 0.4$	PCC																				0	
$Pr(Mut) = 0.0$	Num Cats																					
$Pr(Cat_{del}) = 0.4$	PCC	1	1	1	1	1	1	1	1	1	1	1		1	1		1	1	1	1	17	
$Pr(Mut) = 0.1$	Num Cats																					1
$Pr(Cat_{del}) = 0.4$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1
$Pr(Mut) = 0.2$	Num Cats																					
$Pr(Cat_{del}) = 0.4$	PCC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	
$Pr(Mut) = 0.4$	Num Cats													*								2

is not recommended for GEAM, since it always (for all databases) results in a GEAM that is not performing well. Additionally, a careful observation of the GGAM results indicate that any value of $Pr(Cat_{del})$ in the interval $[0.2, 0.4]$, and any value of the $Pr(Mut)$ in the interval $[0.05, 0.4]$ gives good results for GGAM. Also, the best performing parameter setting for GGAM is $Pr(Cat_{del}) = 0.4$, and $Pr(Mut) = 0.1$, since for this parameter setting we observe the highest number of good performances (19), and best performances (4) of the associated GGAM. Finally, a probability of mutation equal to 0 is not recommended for GGAM, since it always (for all databases) results in a GGAM that is not performing well.

12.4.3 Comparison of GART with other ART architectures

The second objective of our experimentation was to compare GFAM, GEAM, and GGAM (summarily referred to as GART) with other ART architectures that have previously 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, introduced by Anagnostopoulos, et al., 2002, [2], and Verzi, et al., 2001, [35]. 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 Gomez-Sanchez, et al., 2001, [16]) 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 12.3)). 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 12.3.

In particular, the parameter settings that we experimented with ssFAM were: baseline vigilance values ranging from 0 to 0.9 with step size of 0.1, choice

parameter values of 0.001 and 0.01, maximum allowable mixture threshold values ranging from 0 to 1 with step size of 0.1, and 100 different orders of pattern presentations of the training data (resulting in 22,000 different parameter settings). Furthermore, the settings for ssEAM were: baseline vigilance values ranging from 0 to 0.9 with step size of 0.1, choice parameter values of 0.001 and 0.01, maximum allowable mixture threshold values ranging from 0 to 1 with step size of 0.1, minimum axes to maximum axis ratio values ranging from 0.1 to 1 with step size of 0.1, and 100 different orders of pattern presentations of the training data (resulting in 220,000 different parameter settings). Also, the settings for ssGAM were: baseline vigilance values ranging from 0 to 0.9 with step size of 0.1, initial standard deviation parameter ranging from 0.1 to 1 with step size of 0.1, maximum allowable mixture threshold values ranging from 0 to 1 with step size of 0.1, and 100 different orders of pattern presentations of the training data (resulting in 110,000 different parameter settings). Finally, the settings for safe micro-ARTMAP were: baseline vigilance values ranging from 0 to 0.4 with step size of 0.2, baseline vigilance parameter values of 0.001 and 0.01, 5 values for the maximum "all-categories" entropy threshold, 6 different ratios of the values of the "categories" entropy threshold to the "all-categories" entropy threshold, three values of the maximum allowable expansion of a category, and 100 different orders of pattern presentations of the training data (resulting in 90,000 different parameter settings).

The best parameter setting, identified in the previous sub-section, 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 12.3. The results shown in Table 12.3 are truncated to one decimal place.

12.4.4 Observations from the results

Some of the conclusions that can be deduced from the comparative results, depicted in Table 12.3, are emphasized below.

- **Observation 1 (Overall Performance of GART networks):** GFAM, GEAM and GGAM attain good performance on all the datasets, and quite often, optimal performance (e.g., see performance of all the networks in the Gaussian databases, and performance of GGAM on the structures-within-structure problems, and on the real databases). The best performing network from the class of GART networks (GFAM, GEAM, and GGAM) is GGAM.
- **Observation 2 (Comparative Performance of GART networks, with respect to each other)**. GGAM and GEAM outperform the performance of GFAM on all the structures, within structure problems. For

Table 12.3. Best results obtained from GFAM, GEAM and GGAM compared to best results obtained from Safe μARTMAP, ssFAM, ssEAM and ssGAM

Dataset	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

all the other problems the differences between GEAM, and GGAM versus GFAM are not statistically significant.

- **Observation 3 (Comparative Performance of GART networks compared with ssFAM):** ssFAM performs as well as the GART networks for the 2-class Gaussian datasets. For all the other datasets at least one (if not all) the GART networks perform better (achieving higher PCC with fewer ART categories). The largest difference in PCC observed is almost 12% (for the 4 Circle in the Square problem), while the largest ratio of number of ssFAM versus GART categories is for the modified IRIS problem (ratio of 4).
- **Observation 4 (Comparative Performance of GART networks compared with ssEAM):** ssEAM performs as well as the GART networks for the 2-class Gaussian datasets. For all the other datasets at least one (if not all) the GART networks perform better (achieving higher PCC with fewer ART categories). The largest difference in PCC observed is more than 10% (for the 6 class Gaussian problem with 40% overlap), while the largest ratio of number of ssEAM versus GART categories is for Circle in the Square problem (ratio of 4).
- **Observation 5 (Comparative Performance of GART networks compared with ssGAM):** ssGAM performs as well as the GART networks for the 2-class Gaussian datasets. For all the other datasets at least

one (if not all) the GART networks perform better (achieving higher PCC with fewer ART categories). The largest difference in PCC observed is more than 8% (for the 1 Circle in the Square problem), while the largest ratio of number of ssGAM versus GART categories is for the four Gaussian dataset with 25% overlap problem (ratio larger than 5).

- **Observation 6 (Comparative Performance of GART networks compared with safe micro-ARTMAP):** Safe micro-ARTMAP performs as well as the GART networks for the 2-class, and 4-class Gaussian datasets. For all the other datasets at least one (if not all) the GART networks perform better (achieving higher PCC with fewer ART categories). The largest difference in PCC observed is more than 6% (for the 6 class Gaussian dataset with 25% overlap), while the largest ratio of number of safe micro-ARTMAP versus GART categories is for the Circle in the Square problem (ratio of 4).

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), compared to the large number of networks trained in the ssFAM, ssEAM, ssGAM or micro-ARTMAP cases (at least 22,000). Furthermore, in GFAM, GEAM and GGAM cases we evolved the trained networks $Gen_{max} = 500$ times, each evolution requiring cross-validating $Pop_{size} = 20$ networks. Hence, the total number of networks cross-validated in the ssFAM, ssEAM, ssGAM and micro-ARTMAP cases were at least 22,000, while in the GFAM, GEAM and GGAM networks were 10,000; furthermore the networks cross-validated in the ssFAM, ssEAM, ssGAM, and micro-ARTMAP cases have higher number of category nodes than the ones cross-validated in the GFAM, GEAM and GGAM cases. As a result, we can argue that the improved performance (smaller number of nodes and better generalization) of GFAM, GEAM, and GGAM, compared with ssFAM, ssEAM, ssGAM, and micro-ARTMAP, is achieved with reduced computational effort.

12.5 Conclusions

Adaptive Resonance Theory (ART) neural networks have been introduced into the literature by Carpenter, Grossberg and their colleagues at Boston University, as well as other researchers in the field. The consensus with ART

networks is that they converge fast to a solution for arbitrary classification problems they can provide explanations for the answers that they produce, they can function in an on-line training mode, and they solve effectively a variety of classification problems. However, all these benefits sometimes come at the expense of unnecessarily creating (at times) too many categories to solve the problem at hand, referred to as the category proliferation problem in ART. This problem is more acute when ART is confronted with classification problems that deal with noisy or highly overlapping data. To alleviate this problem a number of researchers have proposed solutions such as ssFAM, ssEAM (see Anagnostopoulos, et al., 2002 and 2003, [2, 3], and Verzi, et al., 2001, [35]), ssGAM (see Chalasani, 2005 [12]), Safe micro-ARTMAP (see Gomez, et al., 2001, [16]), dFAM (Carpenter, et al., 1998, [11]), FasART, and dFasART (Parado-Hernandez, et al., 2003, [30]), to mention only a few.

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.

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). The ART network identified at the end of the evolutionary process (last generation) was the FAM, EAM or GAM network that attained the highest fitness value (referred to as GFAM, GEAM, or GGAM, respectively). 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** (see our observations in earlier sections).

Furthermore, GART networks were found to be superior to a number of other ART networks (ssFAM , ssEAM , ssGAM , safe micro-ARTMAP) that have been introduced into the literature to address the category proliferation problem in ART. More specifically, GFAM , 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. More specifically, in some instances the difference in classification performance of GFAM, GEAM, and GGAM with these other ART networks quite significant (as high as 12%). Also, in some instances the ratio of the number of categories created by these other ART networks, compared to the categories created by GFAM, GEAM or GGAM was large (as high as 5).

Acknowledgment This work was supported in part by the National Science Foundation (NSF) grant CRCD: 0203446, and the National Foundation grant DUE 05254209. Georgios Anagnostopoulos and Michael Georgiopoulos acknowledge the partial support from the NSF grant CCLI 0341601.

References

1. Anagnostopoulos GC (2001) Novel Approaches in Adaptive Resonance Theory for Machine Learning. PhD Thesis, Univ Central Florida, Orlando
2. Anagnostopoulos GC, Georgiopoulos M, Verzi SJ, Heileman GL (2002) Reducing generalization error and category proliferation in ellipsoid ARTMAP via tunable misclassification error tolerance: boosted ellipsoid ARTMAP. In: Proc IEEE-INNS-ENNS Int J Conf Neural Networks (IJCNN) 3:2650–2655
3. Anagnostopoulos GC, Bharadwaj M, Georgiopoulos M, Verzi SJ, Heileman GL (2003) Exemplar-based pattern recognition via semi-supervised learning. In: Proc IEEE-INNS-ENNS Int J Conf Neural Networks (IJCNN) 4:2782–2787
4. Bala J, De Jong K, Huang J, Vafaie H, Wechsler H, (1996) Using learning to facilitate the evolution of features for recognizing visual concepts. Evolutionary Computation 4:297-311
5. Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and Regression Trees. Wadsworth
6. Brotherton TW, Simpson PK (1995) Dynamic feature set training of neural nets for classification. In: McDonnel JR, Reynolds RG, Fogel DB (eds) Evolutionary Programming IV pp 83–94. MIT Press, Cambridge, MA
7. Burton AR, Vladimirova T (1997) Utilisation of an adaptive resonance theory neural network as a genetic algorithm fitness evaluator. In: Proc IEEE Int Symp Information Theory pp 209
8. Cantu-Paz E (2002). Feature sub-set selection by estimation of distribution algorithms. In: Langdon WB, Cantu-Paz E, Mathias K, Roy R, Davis D, Poli R, Balakrishna K, Honavar V, Rudolph G, Wegener J, Bull L, Potter MA, Schultz AC, Miller JF, Burke E, Jonoska N (eds) Proc Genetic and Evolutionary Computation Conf pp 303–310. Morgan Kaufmann Publishers, San Francisco, CA
9. Carpenter GA, Grossberg S, Reynorlds JH (1991) Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks 4:759–771
10. Carpenter GA, Grossberg S, Markuzon N, Reynolds JH, Rosen DB (1992) Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Transactions on Neural Networks 3:698–713
11. Carpenter GA, Milenova B, Noeske B (1998) dARTMAP: a neural network for fast distributed supervised learning. Neural Networks 11:793–813
12. Chalasani R (2005) Optimization of Network Parameters and Semi-supervision in Gaussian ART Architectures. MS Thesis, Univ Central Florida, Orlando
13. Charalampidis D, Kasparis T, Georgiopoulos M (2001) Classification of noisy signals using fuzzy ARTMAP neural networks. IEEE Transactions on Neural Networks 12:1023–1036
14. Georgiopoulos M, Huang J, Heileman GL (1994) Properties of learning in ARTMAP. Neural Networks 7:495–506

15. Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA
16. Gomez-SanchezE, Cano-Izquierdo JM, Lopez-Coronado J (2001) Safe-uARTMAP: a new solution for reducing category proliferation in fuzzy ARTMAP. In: Proc Int J Conf on Neural Networks
17. Gomez-Sanchez E, Dimitriadis YA, Cano Izquierdo JM, Lopez-Coronado J (2002) uARTMAP: use of mutual information for category reduction in Fuzzy ARTMAP. IEEE Transactions on Neural Networks 13:58–69
18. Grossberg S (1976) Adaptive pattern classification and universal recoding, II: feedback, expectation, olfaction, and illusions. Biological Cybernetics 23:187–202
19. Hancock PJB (1992). Pruning neural networks by genetic algorithm. In: Aleksander I, Taylor J (eds) Proc Int Conf on Neural Networks 2:991–994. Elsevier Science, Amsterdam, Netherlands
20. Inza I, Larranaga P, Etxeberria R, Sierra B (2000) Feature sub-set selection by Bayesian networks based optimization. Artificial Intelligence, 123:157–184
21. Kaburlasos VG (2006) Towards a Unified Modeling and Knowledge-Representation Based on Lattice Theory — Computational Intelligence and Soft Computing Applications, ser Studies in Computational Intelligence 27. Springer, Heidelberg, Germany
22. Kaburlasos VG, Athanasiadis IN, Mitkas PA (2007) Fuzzy lattice reasoning (FLR) classifier and its application for ambient ozone estimation. Intl J Approximate Reasoning 45(1):152–188
23. Kasuba T (1993) Simplified fuzzy ARTMAP. AI Expert 18–25
24. Kelly J, Davis L (1991) Hybridizing the genetic algorithm and the k-nearest neighbors classification algorithm. In: Belew RK, Booker LB (eds) Proc Fourth Int Conf Genetic Algorithms pp 377–383
25. Koufakou A, Georgiopoulos M, Anagnostopoulos GC, Kasparis T (2001) Cross-validation in fuzzy ARTMAP for large databases. Neural Networks 14:1279–1291
26. Liu H, Liu Y, Liu J, Zhang B, Wu G (2003) Impulse force based ART network with GA optimization. In: Proc Int Conf Neural Networks and Signal Processing 1:499–502
27. Martin FJM, Hernandez FS (1993) Genetic synthesis of discrete-time recurrent neural networks. In: Proc Int Workshop Artificial Neural Networks (IWANN) LNCS 686:179–184. Springer-Verlag, London, UK
28. Marriott S, Harrison RF (1995) A modified fuzzy ARTMAP architecture for the approximation of noisy mappings. Neural Networks 8:619–641
29. Newman DJ, Hettich S, Blake, CL, Merz CJ (1998) UCI Repository of machine learning databases `http://www.ics.uci.edu/ mlearn/MLRepository.html` University of California, Irvine, CA
30. Parrado-Hernadez E, Gomez-Sanchez E, Dimitriadis YA (2003) Study of distributed learning as a solution to category proliferation in fuzzy ARTMAP based neural systems. Neural Networks 16:1039–1057
31. Petridis V, Kaburlasos VG (1998) Fuzzy lattice neural network (FLNN): a hybrid model for learning. IEEE Transactions on Neural Networks 9: 877–890
32. Sexton RS, Dorsey RE, Jonson JD (1998) Toward global optimization of neural networks: a comparison of the genetic approach and back-propagation. Decision Support Systems 22:171–185
33. Siedlecki W, Sklansky J (1989) A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters 10:335–347

34. Taghi M, Bagmisheh V, Pavesic N (2003) A fast simplified fuzzy ARTMAP network. Neural Processing Letters 17:273–316
35. Verzi SJ, Heileman GL, Georgiopoulos M, Healy, M (2001) Rademacher penalization applied to fuzzy ARTMAP and boosted ARTMAP. In: Proc IEEE-INNS Int J Conf on Neural Networks (IJCNN) pp 1191–1196
36. Williamson JR (1996) Gaussian ARTMAP: a neural network for fast incremental learning of noisy multidimensional maps. Neural Networks 9:881–897
37. Williamson JR (1997) A constructive, incremental-learning network for mixture modeling and classification. Neural Computation 9:1517–1543
38. Yang J, Honavar V (1998). Feature subset selection using genetic algorithms. IEEE Intelligent Systems 13:44–49
39. Yao X, Liu Y (1998) Making use of population information in evolutionary artificial neural networks. IEEE Transactions on Systems, Man, Cybernetics part B 28:417-425
40. Yao X (1999) Evolving artificial neural networks. Proceedings of the IEEE 87(9):1423–1447

Chapter 12 Appendix

GFAM: It denotes the genetically engineered Fuzzy ARTMAP architecture.
GEAM: It denotes the genetically engineered Ellipsoidal ARTMAP architecture.
GGAM: It denotes the genetically engineered Gaussian ARTMAP architecture.
GART: It is a generic name referring to the genetically engineered ART architectures, as a whole, such as GFAM, GEAM, and GGAM.
ssFAM: It denotes the semi-supervised Fuzzy ARTMAP architecture. For more details see [2, 3].
ssEAM: It denotes the semi-supervised Ellipsoidal ARTMAP architecture. For more details see [2, 3].
ssGAM: It denotes the semi-supervised Gaussian ARTMAP architecture. For more details see [12].
Safe micro-ARTMAP: It the ART architecture introduced by Gomez-Sanchez, et al., in 2001 (see [16]).
$\bar{\rho}_a$**; Baseline Vigilance Parameter:** One of the parameters of the Fuzzy ARTMAP (FAM), Ellipsoidal ARTMAP (EAM) and Gaussian ARTMAP (GAM) architectures that controls the size of the categories created in FAM, EAM and GAM.
Choice Parameter: One of the parameters of the Fuzzy ARTMAP (FAM) and Ellipsoidal ARTMAP (EAM) architectures that controls the value of bottom-up input of a category in FAM or EAM.
γ**; Initial Standard Deviation Parameter:** One of the parameters of the Gaussian ARTMAP architecture that controls the initial width of the Gaussian probability distribution of a GAM category.

μ**; Minor Axes to Major Axis Parameter:** One of the parameters of the Ellipsoidal ARTMAP architecture that defines the ratio of the minor axes to major axis of the ellipsoidal categories in the EAM architecture.
m; Center of a Category: One of the parameters that describes a category in EAM or GAM. In EAM it corresponds to the center of the ellipsoidal category, while in GAM it corresponds to the center of the Gaussian probability distribution, representing a GAM category.
d; Direction Vector of a Category: One of the parameters that describes a category in EAM. It corresponds to the direction of the major axis of the ellipsoid.
r**; Radius of a Category:** One of the parameters that describes a category in EAM. It is equal to the half length of the major axis.
σ**; Standard Deviation Vector of a Category:** One of the parameters that describes a category in GAM. The components of this vector define the standard deviation of the Gaussian distribution across every dimension of the input pattern space.
n**; Number of Patterns Encoded by a Category:** One of the parameters that describes a category in GAM.