FAST, STABLE AND ONLINE TRAINING OF FUZZY ARTMAP USING A NOVEL, CONSERVATIVE, SLOW-LEARNING STRATEGY

K. Carr *, K. Cannava*, R. Pescatore*, M. Georgiopoulos*, G. C. Anagnostopoulos**
* Department of ECE, University of Central Florida, Orlando, FL 32826
* Department of ECE, Florida Institute of Technology, Melbourne, FL 32901

ABSTRACT
In this paper we present a novel type of an on-line learning ART neural network algorithm. ART neural network architectures have been used successfully for classification purposes and feature a number of desirable characteristics, such as: on-line learning capabilities, stable learning, recognition of novel inputs, and ease of explaining the answers that they produce, among others. In the ART algorithm that we propose, we introduce a new learning strategy encompassing a new, *slow-learning* rule and a new type of *match tracking*, according to which category updates or creations are reduced to a minimum to control the proliferation of categories, while maintaining the on-line and stable learning characteristics of the original ART architectures. We have conducted a number of experiments on artificial and real databases to demonstrate the advantages of our proposed ART neural network algorithm.

1. INTRODUCTION

NEURAL networks have been used extensively and successfully to address a wide variety of problems. The ones that are of interest to us in this work are the class of neural network architectures referred to as ART neural networks. ART neural networks are based on the adaptive resonance theory developed by Grossberg (see Grossberg, 1976). Two of the ART neural network architectures that are important in this work is the Fuzzy ARTMAP neural network (see Carpenter, et al., 1992) and the class of semi-supervised ART neural network architectures discussed in Anagnostopoulos, et al., 2002b. Fuzzy ARTMAP is considered the benchmark supervised ART neural network architecture. Semi-supervised versions of ART were introduced (see Anagnostopoulos, et al., 2003) to remedy the category proliferation problem in ART and to eliminate the zero post-training error that Fuzzy ARTMAP exhibits. The category proliferation problem is the problem encountered in the original ART architectures where many ART categories were unnecessarily created in the cases where ART was exposed to noisy and/or overlapping data. Remedies to the category proliferation problem in ART have been suggested by a variety of researchers (Williamson, 1996; Verzi, et al., 1998; Gomez-Sanchez, et al., 2002; and Anagnostopoulos et al., 2002b, and 2003).

The category proliferation problem was the main focus for the modification of FAM and ssFAM. We refer to these modifications as *stable semi-supervised FAM (sssFAM)*. A category, whose definition will become clearer later, is essentially a representation of a cluster of the input data. In ART we wish to keep the numbers of the categories and their corresponding sizes to a minimum. Our contribution to the ART algorithms is an attempt to decrease the amount of categories created and the size of these categories. We expand upon the modifications that were made with the semi-supervised FAM, and implement a slow-learning approach to category updates as well as the opportunity for minimum updates or no updates at all. The organization of this document is as follows. In section 2 we provide the necessary background about Fuzzy ARTMAP (FAM) and semi-supervised Fuzzy ARTMAP (ssFAM). In section 3 we describe the special features of sssFAM. In section 4 we elaborate on the experimental results conducted that allows us to compare sssFAM versions and ssFAM. Finally, in section 5 we summarize our work and we emphasize some concluding remarks.

2. THE FAM ARCHITECTURE --- FAM AND SSFAM

The Fuzzy ARTMAP architecture consists of four layers or fields of nodes (see Figure 1). The layers that are worth describing are the *input layer* F_1^a, the *category representation* layer F_2^a, and the *output layer* F_2^b. The input layer of Fuzzy ARTMAP is the layer where an input vector of dimensionality $2M_a$ of the following form is applied

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1, a_2, \ldots, a_{M_a}, a_a^c, a_2^c, \ldots, a_{M_a}^c) \tag{1}$$

$$a_i^c = 1 - a_i; \forall i \in \{1, 2, \ldots, M_a\} \tag{2}$$

The assumption here is that the input vector $\boldsymbol{a}$ is such that each one of its components lies in the interval [0,1].

The layer F_2^a of Fuzzy ARTMAP is referred to as the *category representation layer*, because this is where categories (or groups) of input patterns are formed. Finally, the output layer is the layer that produces the outputs of the network. An output of the network represents the output to which the input applied at the input layer of FAM is supposed to be mapped to.

There are two sets of weights worth mentioning in FAM. The *first set of weights* are weights from F_2^a to F_1^a, designated as $\mathbf{w}_{ji}^{a}$, $1 \le j \le N_a, 1 \le i \le 2M_a$, and referred to as top-down weights. The vector of weights:

$$\mathbf{w}_j^a = (w_{j1}^a, w_{j2}^a, \cdots, w_{j,2M_a}^a) \tag{3}$$

is called a *template*. Its functionality is to represent the group of input patterns that chose node *j* in the category representation layer of Fuzzy ARTMAP as their representative. Hence, a template constitutes a compressed representation of all the input patterns that are represented by it. A template in FAM has an interesting geometrical interpretation. It represents a hyperectangle whose boundaries enclose all the input patterns that chose and were encoded by this template. The *second set of weights*, worth mentioning, consists of weights that emanate from every node *j* in the category representation layer to every node *k*

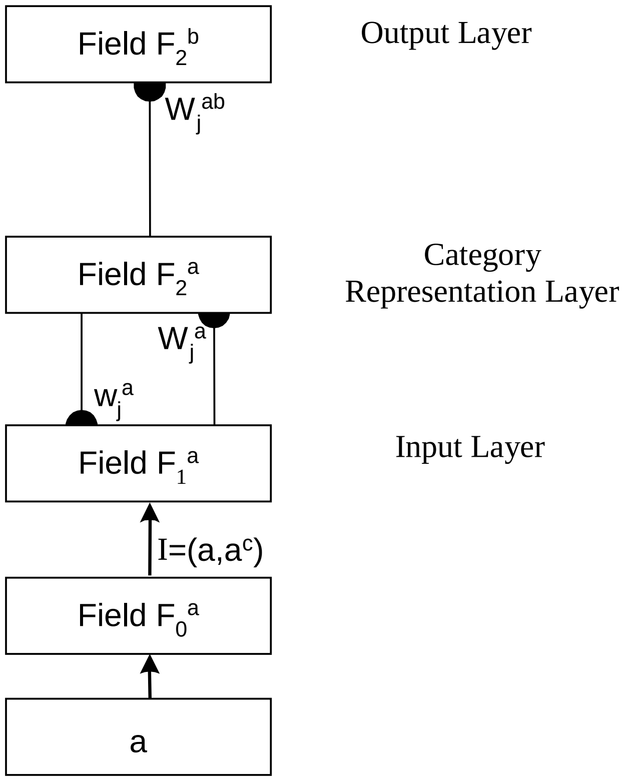

Fig. 1. Block Diagram of Fuzzy ARTMAP Architecture

in the output layer. These weights are designated as W_{jk}^{ab} (called inter--ART weights). The vector of inter--ART weights emanating from every node *j* in FAM, denoted by

$$\mathbf{W}_j^{ab} = \left(W_{j1}^{ab}, W_{j2}^{ab}, \cdots, W_{j,N_b}^{ab}\right) \tag{4}$$

corresponds to the output pattern that this node j is mapped to. The functionality of $\mathbf{W}_j^{ab}$ is to designate the label (output pattern) that the group of input patterns, represented by the template $\mathbf{w}_j^a$, are mapped to.

Fuzzy ARTMAP can operate in two distinct phases: the *training phase* and the *performance phase*. In the training phase of FAM a list of input/output pairs, designated by

$$\left\{\left(\mathbf{I}^1, \mathbf{O}^1\right), \left(\mathbf{I}^2, \mathbf{O}^2\right), \ldots, \left(\mathbf{I}^P, \mathbf{O}^P\right)\right\} \tag{5}$$

is repeatedly presented to FAM until the network learns the required mappings from inputs to outputs. When FAM is used as a classifier, the output patterns $\mathbf{O}^K$ represent class labels. In the performance phase of FAM a list of input patterns, such as $\tilde{\mathbf{I}}^1, \tilde{\mathbf{I}}^2, \cdots, \tilde{\mathbf{I}}^S$, is presented to FAM and the output of the network is recorded.

The operation of FAM is affected by two network parameters, the choice parameter β_a, and the baseline vigilance parameter $\bar{\rho}_a$. The choice parameter takes values in the interval $(0, \infty)$, while the baseline vigilance parameter assumes values in the interval [0,1]. Both of these parameters affect the number of nodes created in the category representation layer of Fuzzy ARTMAP. Higher values of β_a and $\bar{\rho}_a$ create more nodes in the category representation layer of Fuzzy ARTMAP, and consequently produce less compression of the input patterns. There are two other network parameter values in Fuzzy ARTMAP that are worth mentioning. These are the vigilance parameter ρ_a, and the number of nodes N_a in the category representation layer of Fuzzy ARTMAP. The vigilance parameter ρ_a takes value in the interval $[\bar{\rho}_a, 1]$ and its initial value is set to be equal to $\bar{\rho}_a$. The number of nodes N_a in the category representation layer of Fuzzy ARTMAP increases while training the network and corresponds to the number of committed nodes in Fuzzy ARTMAP plus one uncommitted node.

There are three major operations that take place during the presentation of a training input/output pair (e.g., $(\mathbf{I}, \mathbf{O})$) to Fuzzy ARTMAP.

Operation 1 (Calculation of the Category Choice Function (CCF) value): Calculation of the category (node) choice function value (i.e., $T(j \mid \mathbf{I})$) for every node (category) j in F_2^a, is as follows:

$$T(j \mid \mathbf{I}) = \frac{M_a - s(\mathbf{w}_j^a) - dis(\mathbf{I}, \mathbf{w}_j^a)}{\beta_a + M_a - s(\mathbf{w}_j^a)} \tag{6}$$

After calculation of the choice function values the node J with the maximum category choice function value is chosen.
Operation 2 (Calculation of the Category Match Function (CMF) Value): The node J with the largest CCF value is examined to determine whether it passes the vigilance test (VT). A node (category) passes the vigilance test (VT) if its category (node) match function value (i.e., ($\rho(j \mid \mathbf{I})$) exceeds the vigilance parameter value ρ_a, that is if

$$\rho(j \mid \mathbf{I}) = \frac{M_a - s(\mathbf{w}_J^a) - dis(\mathbf{I}, \mathbf{w}_J^a)}{M_a} \geq \rho_a \tag{7}$$

If the vigilance test is passed we proceed with operation 3. Otherwise, node J is disqualified and we find the next in sequence node in F_2^a that maximizes the CCF value. Eventually we will end up with a node J that maximizes the CCF value and satisfies the vigilance test.

Operation 3 (Match Tracking Mechanism/Change of the Weights): This operation is implemented only after we have found a node J that maximizes the CCF value of the remaining (in the competition) F_2^a nodes and passes the vigilance criterion. Operation 3 determines whether this node J passes the prediction test. The prediction test checks if the inter-ART weight vector emanating from node J (i.e., $\mathbf{W}_J^{ab} = (W_{J1}^a, W_{J2}^a, ..., W_{J,N_b}^a)$) matches exactly the desired output vector $\mathbf{O}$ (if it does, this is referred to as the node "passing the prediction test"). If the node does not pass the prediction test, the vigilance parameter ρ_a is increased to the level of $\dfrac{M_a - s(\mathbf{w}_J^a)}{M_a}$, node J is disqualified, and the next in sequence node J that maximizes the CCF value and passes the vigilance is chosen (this action is referred to as match tracking mechanism). If node J though passes the prediction test, the weights $\mathbf{w}_J^a$ in FAM are modified. The weights in FAM are modified in a way that the hyper-rectangle corresponding to $\mathbf{w}_J^a$ is expanded to include within its boundaries the new input pattern. In all of the above equations the quantities $s(\mathbf{w}_j^a)$ and $dis(\mathbf{I}, \mathbf{w}_J^a)$ appear; $s(\mathbf{w}_j^a)$ represents the size of a rectangle and $dis(\mathbf{I}, \mathbf{w}_J^a)$ is the distance of pattern $\mathbf{I}$ from the rectangle corresponding to template $\mathbf{w}_J^a$. Figure 2 illustrates the size of a rectangle corresponding to template $\mathbf{w}_J^a$ and the distance of a pattern $\mathbf{I}$ from this rectangle.

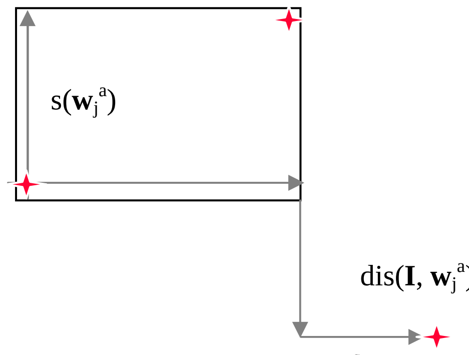

Fig.2 A pictorial illustration of a category (represented by the rectangle $\mathbf{w}_j^a$ with size $s(\mathbf{w}_j^a)$, depicted on the figure), and the distance of a pattern $\mathbf{I}$ from the rectangle $\mathbf{w}_j^a$ designated as $dis(\mathbf{I}, \mathbf{w}_j^a)$ and depicted on the figure.

Notice that in the exposition above we have omitted mentioning the commitment test (CT) for reason of clarity and brevity. More specifically, we have assumed that the weights of all uncommitted nodes are initialized to $w_u \rightarrow \infty$, which disengages the CT from FAM's operation (Anagnostopoulos & Georgiopoulos, 2002).

The ssFAM algorithm relies on the same neural network architecture as the one exhibited in Figure 1. The difference between FAM and ssFAM is that FAM allows no error to occur in the training set, while ssFAM allows a certain amount of training error, which is controlled by a tunable network parameter $\varepsilon \in [0,1]$ called *category prediction error tolerance*. Whenever a category is formed (in both FAM and ssFAM) it is attributed an initial class label which is the label of the first pattern that accessed this category. In ssFAM a category is guaranteed that it will not exceed a prediction error of $100\varepsilon\%$ with respect to the initial class label. It is worth pointing out that ssFAM stores category-class label associations in the weights W_{jk}^{ab}; for example W_{jk}^{ab} being equal to 3 indicates that three training input patterns with label k were encoded by or have accessed node j in the category representation layer during the training phase of ssFAM. The test that ssFAM employs to determine if the post-training error of a category *j* is allowable is shown below.

$$\frac{W_{j,L_0(j)}^{ab}}{1+\sum_{c=1}^{C} W_{jc}^{ab}} < 1-\varepsilon \qquad (8)$$

In the above equation, the numerator represents the number of patterns of the initial label encoded by the category and the denominator is the total patterns that the category has already encoded plus one (representing the new pattern that is examining of whether this test would allow it to be encoded by this category). It has been shown that ssFAM improves both the number of categories and the percentage of correct classification (Anagnostopoulos, et al., 2003). In the following figure we show graphically how ssFAM functions by allowing patterns of different labels to be encoded by the same category. As Figure 3 shows, the category under consideration is encoding seven patterns of the initial label, (red color), and three of a different label (blue color). We assumed here that the patterns of the initial label (red) were encoded first and then three patterns of a different label (blue) were allowed to be encoded because ε was chosen to be equal to 0.3.

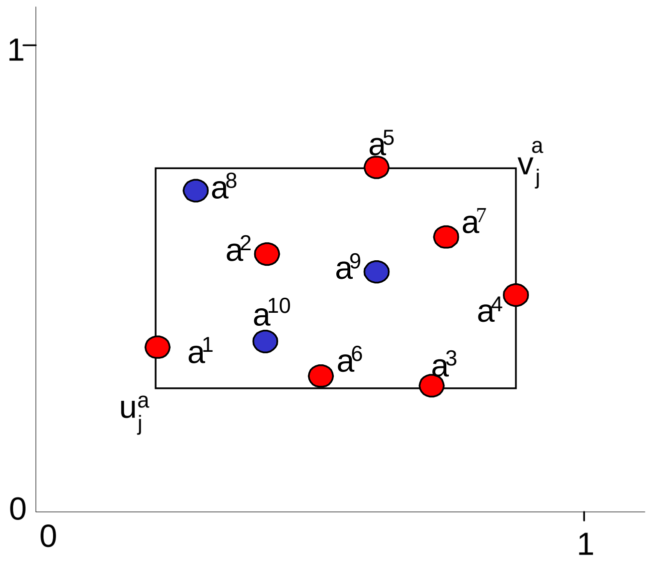

Fig. 3. A rectangle (category) in ssFAM showing patterns with two different labels (denoted by the different colors) encoded by the category

3. STABLE ssFAM (sssFAM)

Let us first present the pseudo-code of sssFAM and then we will elaborate on the features that make it different than the FAM and ssFAM counterparts.

Choose a training pattern

Calculate the CCF values of every node in the category representation layer; Find the node with the maximum CCF value
Calculate the CMF value of the node with (maximum CCF value; Perform the vigilance test
Find the winning node (node with maximum CCF value that passes the vigilance test)

```
Perform the Prediction Test for the winning node

    If the winning node passes the prediction test (PT)
        Perform new threshold vigilance test; if it passes update via fast learning; otherwise do nothing

    If the winning node fails the prediction test
        Perform new match tracking (i.e., temporarily raise the vigilance value and perform the new vigilance test
        for the category with the same label as the input pattern and of maximum CMF value)
        If the category, thus chosen, passes the vigilance test and the new threshold vigilance test update via fast
        learning; otherwise update via slow learning
```

The elements of the pseudo-code that were underlined are worth elaborating more and they are the elements that are distinguishing sssFAM from FAM, ssFAM.

Let us first focus on the new match tracking rule. According to the old match tracking rule utilized in FAM and ssFAM, if a category of the wrong label is accessed, the vigilance is increased and another category that maximizes the CCF value and passes the new vigilance is examined. In turn, this category might be of the wrong label and the match tracking process is invoked once again. Stable ssFAM (sssFAM) avoids these repeated invocations of the match tracking rule by immediately examining, after the first match tracking pass, the node with the maximum CMF value, whose label is the same as the label of the input pattern. This guarantees that only one match tracking pass will be necessary.

Our second focus is the new vigilance threshold test. To facilitate this additional test we introduced an additional parameter in sssFAM, called r, ranging in the interval [0, 1]. The threshold vigilance is calculated using this r parameter, which in turn affects the maximum size a category is allowed to have. In particular, a comparison is made between the category's vigilance ratio and this threshold vigilance, as indicated below.

$$\rho(J \mid \mathbf{x}) \geq \bar{\rho}_{thres} = 1 - r(1 - \bar{\rho}) \tag{9}$$

If the vigilance ratio of the category with the correct label is greater than the threshold vigilance, an update using fast learning will occur; otherwise no update of the category occurs. An equivalent way of looking at this threshold vigilance test (and easier to understand) is demonstrated below

$$s' \leq M(1 - \bar{\rho}_{thres}) \tag{10}$$

This equivalent test states that if the size of category s' with the correct label, when updated, is less than or equal to the right hand side of the above equation then perform an update via fast learning; otherwise perform no update of the category. What is interesting about this equivalent test is that the right hand side of the above equation represents size, which we will call s_{thresh}. In essence, if the updated category's size does not exceed a certain size limit, controlled via the r parameter, then perform fast learning. Otherwise, do not perform an update.

In the case, where match tracking has already been invoked, this new threshold vigilance check is performed in the exact same fashion. However, instead of deciding whether or not to update the category, we are deciding of whether or not to update it using slow learning with a minimum learning rate calculation or to update it using fast learning. Once more, we update using fast learning if the new vigilance threshold test is passed and we update it using slow learning (with a minimum learning rate) if the new vigilance threshold test is violated.

The minimum learning rate calculation is relatively straightforward. Consider the situation where match tracking has been invoked in sssFAM and T_J is the CCF value of the category of the incorrect label, while $T_{J'}$ is the CCF value of the category with maximum CMF value and of the correct label, which we are going to update. It can be shown that

$$\gamma_{\min} = \left(1 - \frac{T_{J'}}{T_J}\right) \frac{\left[M_a - s(w_{J'}^a) + \beta_a\right]}{dis(w_{J'}^a, I)} \tag{11}$$

This minimum learning rate is the minimum learning that needs to be applied to category J' so that when the same input pattern is immediately presented again, category J' will have the highest CCF value of all other categories (including category J of course). Once this minimum learning rate is calculated we update the category J' using slow learning with a learning rate:

$$\gamma = (1 - \zeta)\gamma_{\min} + \zeta \tag{12}$$

The parameter ζ is the nominal learning rate, set before training. Slow learning corresponds to the situation where the rectangle that corresponds to the category that it is updated, due to the presentation of a new input pattern, does not expand to its full extent (to incorporate within its boundaries the new input patterns).

Before experimentation began, two different versions of stable ssFAM were considered. Stable ssFAM version 1 (sssFAM v.1) is the algorithm described above. On the other hand, version 2 differs from version 1 as follows: it allows updates via fast learning if the threshold vigilance test passes and if the category updated is a point category. In that sense, version 2 is slightly more conservative version of version 1.

4. EXPERIMENTAL RESULTS

In order to show the advantages of sssFAM (versions 1 and 2) we conducted a collection of experiments using artificially generated data sets, as well as real databases. We separated the data of a database into three distinct sets: a training set, a cross-validation set and a test set. The training set was used to train the neural network architectures, the cross-validation set was used to find the "optimum" network parameter values, and the test set was used to produce the performance results of the networks with the "optimum" network parameter values. The "optimum" network parameter values were defined to be the ones that attained the 100 best generalization performances of the trained network on the cross-validation set. Justifiably, the resulting networks were referred to as the "best" networks. To find the "best" networks we experimented with various values of the network parameters, such as baseline vigilance, choice parameter, error tolerance and orders of pattern presentation (for ssFAM), and all of these parameters plus the two additional parameters for sssFAM version 1 and 2 (r and ζ). For the 100 "best" generalization-performing networks, we also identified the number of categories created. Then we created plots that depicted in 2-D the PIC (Percentage of Incorrect Classification) and NC (number of categories created) of the best 100 networks for each one of the ssFAM, sssFAM (v.1) and sssFAM (v.2) algorithms. Obviously, algorithms for which the 2-D points are closer to the origin are the best performing algorithms (corresponding to low percentages of incorrect classification and low number of categories created).

Artificial Databases

The advantage of using artificial databases is that we can generate as many training, cross-validation, and test data, as we desire. Working with enough cross validation data-points allowed us to safely state that, the best (with respect to generalization) network parameter values found are indeed optimal. The other advantage of the artificial databases is that we can experiment with different input domain dimensionalities, the number of output classes and the amount of inter-class overlap. In this paper we kept the dimensionality of the input patterns in the simulated data fixed, and equal to 2, and we experimented with the number of output classes (2, 4, or 6) and the amount of overlap amongst data belonging to different classes (overlap values of 5%, 15%, 30% and 40% were used). The artificial databases consisted of Gaussianly-distributed data. Input patterns were drawn from either 2 or 4 or 6 classes. Qualitatively the overlap can be classified as low (5%), medium (15%) or high (30% or 40%). The amount of overlap of the simulated data was defined to be the error rate of the optimum Bayes classifier pertinent to the data. For instance, if the Bayes classifier for a simulated data set exhibited a misclassification rate of x%, then the amount of overlap corresponding to this data-set was defined to be x%. For each one of the above 12 sets of artificial data (3 different number of classes × 4 different degrees of class overlap) we generated a training set, a validation set and a test set of 500, 5,000 and 5,000 data-points, respectively.

Our results show a definite improvement of sssFAM (versions 1 and 2) over ssFAM for 2,4,6-class for the Gaussian databases. For instance, Figure 4 shows the performance of the best 100 networks for ssFAM, sssFAM (v.1) and sssFAM (v.2) on a 4-class problem with 40% overlap. The black circles represent the original ssFAM neural network, the red x's show version 1 of Stable ssFAM, while the blue crosses show version 2. The y-axis on the graph represents the percentage of incorrect classification (PIC) on the test set, while the x-axis represents the number of categories created. As it can be seen from Figure 4 both new FAM versions (sssFAM (v.1) and sssFAM (v.2)) show a definite improvement over ssFAM, exhibiting lower percentage of incorrect classification and creating less categories. In particular, as we see from Figure 4 the number of categories created by the sssFAM versions (best 100 networks) are mostly located in the interval of (0, 40) categories, with very few outliers. On the other hand, the number of categories created by ssFAM are mostly located within the interval of 920, 40) categories. More specifically, the average number of categories for the best 100 networks created by sssFAM (v.1), sssFAM (v.2) and ssFAM are 24, 30 and 42, respectively. In a similar fashion, the PIC classification exhibited by the best 100 networks for the sssFAM versions lie in the range of (31.5 32.5). On the contrary the PIC attained by ssFAM (best 100 networks) are within the interval (32 to 34). Actually the average PIC for the best 100 networks of sssFAM (v.1), sssFAM (v.2), and ssFAM are 32%, 32.1%, and 33.1%, respectively. Similar conclusions are valid for the 2-class and 6-class Gaussian problems and other amounts of overlap.

Real Databases

To verify the results obtained with the artificial databases, we conducted similar experiments using a real database (Abalone). This database was chosen from the UCI Machine Learning repository (site…) to test the credibility of the proposed algorithms. The training data set consists of 500 data points selected from the database in a way that reflects the distribution of classes in the database. The selection of the training points within each class is however arbitrary. The number of points remaining in the database, after extracting the training data set, is equally divided to form the test data set and the cross validation data set. The networks created after training are cross-validated with the pertinent network parameters, mentioned earlier. After cross-validation over all the pertinent parameters the best 100 networks are chosen (i.e., the networks exhibiting the best generalization performance on the cross-validation set). Then, as it was the case with the artificial datasets, we created plots that depicted in 2-D the PIC (Percentage of incorrect classification) and NC (number of categories created) of the best 100 networks for each one of the ssFAM, sssFAM (v.1) and sssFAM (v.2) algorithms. Obviously, algorithms for which the 2-D points are closer to the origin are the best performing algorithms (corresponding to low percentages of incorrect classification and low number of categories created).

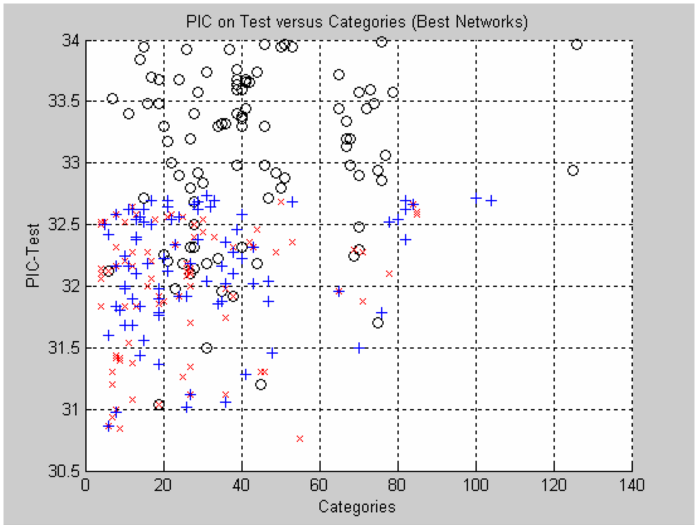

Fig. 4. The Percentage of incorrect classification (PIC) versus number of Categories obtained from the best 100 ssFAM, sssFAM (v.1), and sssFAM (v.2) networks on a 4-class, 40% overlap Gaussian data

The Abalone database pertains to predicting the age of abalone shells from physical measurements. This is pursued to avoid an alternate, more cumbersome method, to do the same. The database has 8 attributes representing the sex, length, diameter, height, weight etc., and consists of 4177 data points. In our experiments, we used 1000 training patterns, 2133 cross validations patterns and 1044 test patterns. The number of rings in the shell of the abalone indicates the age. The database was treated as a classification problem by grouping the number of rings into three categories (1-8, 9-10, and 11-greater) and using the three groups as output classes. Experiments with the Abalone database have been conducted using the aforementioned grouping of categories strategy. In our experiments, the first attribute representing the sex of the abalone was discarded since it was non-numerical. We conjecture that the Abalone dataset has high overlap among its output classes since the best generalization results reported in the literature are around 60%.

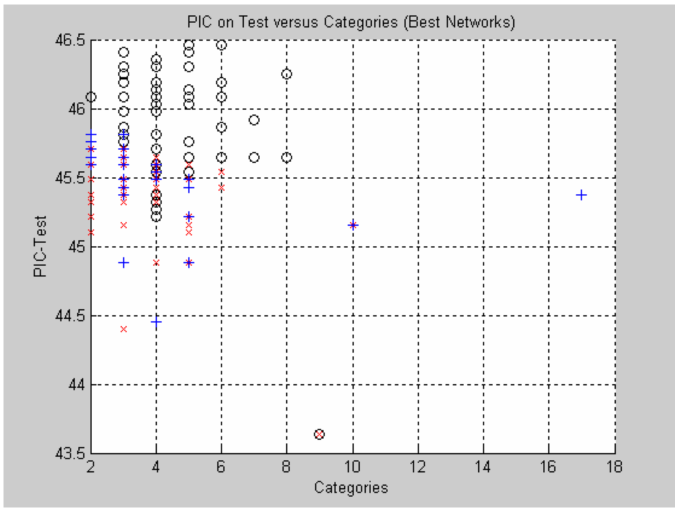

Fig. 5. The Percentage of incorrect classification (PIC) versus number of Categories obtained from the best 100 ssFAM, sssFAM (v.1), and sssFAM (v.2) networks on the Abalone database data

The results obtained with the Abalone database were similar with the results observed with the Gaussian databases. Figure 5 shows the graph of the best 100 networks for each of the versions of FAM (ssFAM, sssFAM (v.1) and sssFAM (v.2)). The best 100 networks for stable ssFAM showed improvement in both the percent correct classification as well as the number of categories created. In particular, the average PIC of the best 100 networks created by sssFAM (v.1), sssFAM (v.2) and ssFAM were found to be 45.35%, 45.5%, and 45.9%, respectively. Furthermore, the average number of categories of the best 100 networks created by sssFAM (v.1), sssFAM (v.2) and ssFAM were 3.6, 3.35, and 4.6, respectively. Although the improvement in generalization performance attained by the sssFAM versions, compared to ssFAM, is minimal, one could not but notice that the sssFAM versions reduced the average number of categories created, despite the excellent ssFAM performance. To be more specific the minimum number of categories that a network could create without sacrificing generalization performance on the Abalone data (3-class problem) is 3. The average number of categories created by the best ssFAM network is 4.6 (very close to 3), but still sssFAM (v.1) manages to reduce this average number by almost a category resulting in a 25% reduction.

5. SUMMARY-CONCLUSIONS

We introduced a new ART algorithm named stable ssFAM (sssFAM). These algorithms were created under the premise of expanding the ART categories by the least required amount, thus avoiding making assumptions about the data where data do not exist. In a way we expected that this sssFAM property would avoid the category proliferation problem observed in ART architectures. We have performed experiments to compare sssFAM versions with the ssFAM counterpart. The experiments were performed on simulated data and on real data. The major conclusion from this comparison is that sssFAM creates networks with fewer categories than ssFAM, and it improves the observed generalization performance slightly. Hence, we believe that there is a definite value to the new ART algorithms that we introduced.

Acknowledgment: Michael Georgiopoulos acknowledges the support from the CRCD NSF grant 0203446 and Georgios C. Anagnostopoulos acknowledges the partial support from the CCLI-EMD NSF grant 0341601.

REFERENCES

[1] Anagnostopoulos, G.C., and Georgiopoulos, M., “Category Regions as New Geometrical Concepts in Fuzzy ART and Fuzzy ARTMAP,” *Neural Networks*, 15(10), pp. 1205-1221, 2002.

[2] Anagnostopoulos, G. C., Georgiopoulos, M., Verzi, S., and Heileman, G. L., “Reducing generalization error and category proliferation in ellipsoid ARTMAP via tunable misclassification error tolerance: Boosted Ellipsoid ARTMAP, ” *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN 2002),* May 12-17, Honolulu, Hawaii, 2002b.

[3] Anagnostopoulos, G. C., Bharadwaj, M., Georgiopoulos, M., Verzi, S. J., Heileman, G. L., “Exemplar-based pattern recognition via semi-supervised learning,” *International Joint Conference on Neural Networks (IJCNN 2003),* Portland, Oregon, July 20-24, Volume 4, pp. 2782-2787, 2003.

[4] Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., “Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multi-dimensional maps,” *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 698-713, 1992.

[5] Gomez-Sanchez, E.; Dimitriadis, Y.A.; Cano-Izquierdo, J.M.; Lopez-Coronado, J., “µARTMAP: use of mutual information for category reduction in Fuzzy ARTMAP”, *IEEE Transactions on Neural Networks*, Volume: 13, Issue:1, pp. 58-69, Jan. 2002.

[6] Grossberg, S., “Adaptive pattern recognition and universal recoding II: Feedback, expectation, olfaction, and illusions,” *Biological Cybernetics*, 23, pp. 187-202, 1976.

[7] Verzi, S.J., Heileman, G.L., Georgiopoulos, M., Healy, M.J., “Boosted ART and Boosted ARTMAP,” *Neural Networks Proceedings*, Vol. 1, 4-9 May 1998 pp. 396-401.

[8] Williamson, J. R., “Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multi-Dimensional Maps,” *Neural Networks*, Vol. 9, No. 5, pp. 881-897, 1996.