Reducing Generalization Error and Category Proliferation in Ellipsoid ARTMAP via Tunable Misclassification Error Tolerance: Boosted Ellipsoid ARTMAP

Georgios C. Anagnostopoulos
School of EE & CS, University of Central Florida
Orlando, FL 32816, USA
and
Michael Georgiopoulos
School of EE & CS, University of Central Florida
Orlando, FL 32816, USA
and
Stephen J. Verzi
Department of EE & CpE, University of New Mexico
Albuquerque, NM 87131, USA
and
Gregory L. Heileman
Department of EE & CpE, University of New Mexico
Albuquerque, NM 87131, USA

Abstract – In this paper we introduce Boosted Ellipsoid ARTMAP (bEAM), a variant of Ellipsoid ARTMAP, which via a tunable misclassification error tolerance increases the network's resubstitution error and implicitly performs structural risk minimization. bEAM constitutes another example of how modifications to Fuzzy ARTMAP can be naturally extended to Ellipsoid ARTMAP.

I. INTRODUCTION

Ellipsoid ARTMAP (EAM) is a neural network architecture rooted on the foundations of Adaptive Resonance Theory (ART) presented by Grossberg [1]. It is a machine that can learn associative mappings from an input domain to an output domain and, as a special case, it can be utilized as a classifier. EAM appeared in [2] as a generalization of Hyper-sphere ARTMAP (HAM) [3]. It is an alternative architecture to Fuzzy ARTMAP (FAM) [4] in the sense that it uses a different geometry for category formation and representation. While FAM constructs and updates categories, whose geometric representation are axis-parallel hyper-rectangles embedded in the input/output domains, EAM employs arbitrarily oriented hyper-ellipsoids to describe distributions of similar data. While FAM measures sizes of categories and distances of patterns from categories via an L_1 metric, EAM accomplishes the same task using weighted L_2 metrics. Using appropriate learning parameter settings, EAM simplifies to HAM and uses L_2-based distances and sizes to cluster relevant data into hyper-spherical categories.

EAM is a successful attempt to derive an ART-based architecture that shares the main characteristics and properties of learning with FAM, while using a different approach to category description. Therefore, by design, EAM is capable of both on-line (incremental) and off-line (batch) learning. Also, an important component of EAM's training and performance phase is the capability of detecting atypical patters, which do not match the structural characteristics of already formed categories. Furthermore, by using local encoding of information, EAM is a *transparent* learning machine, in contrast to *opaque* neural network architectures, for which it is difficult, in general, to explain why an input x produced a particular output y. Additionally, EAM's most interesting and appealing property is the one of self-stabilization in off-line mode via *fast learning* (see Section II.C). Under these circumstances EAM displays fast, finite and stable learning: in a usually small, finite number of *list presentations* (epochs) the network's weights converge, which implies that after a point no categories are being created and already existing categories are not being updated anymore.

However, this desirable characteristic of fast self-stabilization is accompanied by a (rather unwanted, in many cases) side effect: a *resubstitution error* [5] of zero. Stated in a different way, after fast off-line training has completed, EAM has memorized its training set to perfection without committing any errors. Therefore, if EAM is tested on its training set, the percent correct classification will be 100%. Note that by virtue of a zero resubstitution error, the EAM classifier operating in off-line mode via fast learning law belongs to the family of *consistent* [5] classifiers. Especially in the case of abundant noise present in the training data or in the case of highly overlapping classes in the framework of a classification task this type of over-fitting may cause problems to EAM's performance on a test or cross-validation set by affecting its ability to generalize. In this scenario EAM will tend to memorize the noise and will fail to capture the general, underlying characteristics of the training data. In other words, fast off-line learning tends to increase EAM's generalization error to a degree that depends on the noise inherent in the training set. An additional byproduct of memorizing inherent noise is the phenomenon of *category proliferation*, in which EAM will create an increased number of categories in order to cluster appropriately the data and avoid prediction errors.

An ideal approach would be to design an EAM variant that combines the merits of both fast off-line learning and good generalization capabilities. The latter one could be achieved with a design that allows for a resubstitution error larger than zero. A first attempt along this direction has been successfully implemented in Boosted ARTMAP-S (bARTMAP-S) [6]. It is a variant of FAM with a tunable misclassification tolerance parameter that determines the level of the resubstitution error. It has been shown in [6] that bARTMAP-S is capable of reducing its hypothesis complexity (it minimizes *structural risk* [7], [8]) and manages to reduce the gap between resubstitution error and error on a test set, in an indirect fashion.

In this paper we adopt a similar approach to bARTMAP-S and we apply it to EAM in order to improve its generalization ability and reduce the effect of category proliferation when it is trained using fast off-line learning. We call the new architecture Boosted Ellipsoid ARTMAP (bEAM) to indicate its connection to the principals of bARTMAP-S. First, we are going to present an overview of EAM, then we will continue with the description of bEAM and finally we will present some limited, preliminary experimental results, which clearly demonstrate the viability of our approach.

II. ELLIPSOID ARTMAP

In this section we present some important aspects of EAM such as its major components, the way it describes categories, its operation and the way it performs learning.

A. Overview of the Ellipsoid ARTMAP Architecture

EAM consists of two ART modules interconnected via an inter-ART module (also known as *map field*). Each ART module is in essence an Ellipsoid ART (EA) [2] network. While ART_a clusters patterns of the input space, ART_b clusters patterns of a related output space. Moreover, clustering in each module is performed by grouping together similar patterns into *EA categories* (or simply stated, *categories*). The information that describes associations between input and output categories is encoded in the weights w_{jk} of the map field. Each module consists of two layers (fields): the F_1 layer and the representation layer F_2. In contrast to FAM, EA/EAM lack a coding layer F_o, because they do not need to perform complement coding on its input patterns. Both F_1 and F_2 consist of an array of nodes that are interconnected across layers via bottom-up $\mathbf{W}_j$ and top-down weights $\mathbf{w}_j$. The latter ones are also called *templates*. Especially the F_2 layer features two kinds of nodes: committed and uncommitted. The former kind is associated to a template that contains the description of a single category. The latter ones have "blank" templates and correspond to the system's available memory of the system that is used to learn data clusters.

B. Ellipsoid ART Categories

As we have mentioned before, EAM (and EA) summarizes data into groups via the use of categories. In EA/EAM the geometric representation of these categories are hyper-ellipsoids embedded in the data domain. As shown in the Figure 1, each 2-dimensional category *j* corresponds to a committed node *j* and is characterized by a collection of descriptive quantities: a *location (center) vector* $\mathbf{m}_j$, an *orientation vector* $\mathbf{d}_j$ and the length R_j of its major semi-axis (*radius*). The collection of these quantities constitute the node's template $\mathbf{w}_j=[\ \mathbf{m}_j,\ \mathbf{d}_j,\ R_j]$. For uncommitted nodes the templates cannot be defined in this manner, since they do not correspond to "real" categories. The shaded area in Figure 1 is called the *representation region* of category *j* and it encompasses all patterns that the category has encoded.

During the training phase of EA and EAM, learning is accomplished by creating new categories or by expanding already existing ones. A category's template elements are updated incrementally in the light of new evidence provided by the presentation of input patterns. An idiosyncrasy of EA/EAM is that during the training phase the orientation of the hyper-ellipsoids, once decided as suggested by training patterns, will remain fixed despite of potential, future updates. Moreover, during training the ratio of any minor semi-axis length over the length of the major semi-axis is always held constant to a predetermined value $\mu \in (0,1]$, which is common to all categories in the same EA module. It is shown in [9] that exactly this behavior guarantees the fast off-line, self-stabilizing learning property of EA and EAM.

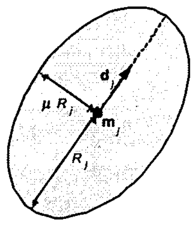

Figure 1: Ellipsoid ART category embedded in a 2-dimensional input space.

Once it is decided that a pattern **x** is going to be encoded into a specific category, in all of these architectures the category will expand enough to include this pattern in its representation region. More precisely, for Fuzzy ART (FA) [10] and FAM the representation region of the updated category will be the minimum hyper-volume, axis-parallel hyper-rectangle that contains both the entire, former representation region and the newly encoded pattern **x**. Similarly, in EA/EAM, Hypersphere-ART (HA) [3] and HAM the updated representation region of an EA (HA) category will be the minimum hyper-volume hyper-ellipsoid (hyper-sphere) that simultaneously contains both the entire, pre-update representation region and the new pattern.

C. Operation of Ellipsoid ARTMAP

EAM has two modes of operation: *training phase* and *performance (testing) phase*. In the first phase the network learns the associations between input domain and output domain categories. Its performance phase is functionally comparable to its training phase with the exception that no categories are updated and no new ones are created. Therefore it suffices to only describe the training phase. Prior to any training all F_2 layer nodes in both $\mathrm{ART_a}$ and $\mathrm{ART_b}$ modules are uncommitted reflecting the fact that the system starts with a blank memory. When training commences, real-valued input-output pairs (**x**,**y**) of training patterns are presented on at a time. In on-line learning mode each pattern is presented to the network only once. In contrast, during off-line learning each pair is presented repeatedly. A single presentation of the complete training set constitutes a *list presentation* (epoch). Thus, off-line learning may involve several list presentations.

During the progress of the training phase some of the F_2 layer nodes may be already committed and correspond to learned clusters. Upon presentation of a pattern **x** to the $\mathrm{ART_a}$ module, all the nodes, committed and uncommitted, will compete for this pattern in terms of category choice function (CCF – or simply, *activation*) values. For a committed node j the CCF value is defined as

$$T(\mathbf{w}_j \mid \mathbf{x}) = \frac{D - R_j - \max\left\{R_j, \left\|\mathbf{x}-\mathbf{m}_j\right\|_{\mathbf{C}_j}\right\}}{D - 2R_j + a} \qquad (1)$$

where

$$\left\|\mathbf{x}-\mathbf{m}_j\right\|_{\mathbf{C}_j} = \frac{1}{\mu}\sqrt{\left\|\mathbf{x}-\mathbf{m}_j\right\|_2^2 - (1-\mu^2)\left[\mathbf{d}_j^{\,T}(\mathbf{x}-\mathbf{m}_j)\right]^2} \quad (2)$$

is a weighted L_2 distance of the pattern **x** from the center of category j. Also, the T-exponent signifies the transpose of the quantity it is applied upon; all vector quantities are assumed to be column vectors. In Equation 1 a>0 is the *choice parameter* and D>0 is the effective diameter of the module's input domain. Usually D is set as

$$D \geq \frac{1}{\mu}\max_{p,q}\left\|\mathbf{x}_p - \mathbf{x}_q\right\|_2 \qquad (3)$$

that is, D is set at least equal to the Euclidian diameter of the input domain divided by μ. This last constraint is there to ensure that CMF values (see Equation 6) remain positive. Uncommitted nodes feature a constant, pattern-independent CCF value of

$$T(\mathbf{w}_j \mid \mathbf{x}) = \frac{D}{2D\omega + a} \qquad (4)$$

where the parameter ω is chosen as $\omega \geq 0.5$ to ensure stability of EAM. The node J featuring the maximum CCF value is considered to be the winner of the winner-take-all competition.

$$J = \arg\max_j T(\mathbf{w}_j \mid \mathbf{x}) \qquad (5)$$

Next, EAM measures the degree to which **x** matches the characteristics of the category corresponding to node J. This is established via the *vigilance test* (VT), which is a major component of EAM's match-based learning and which is depicted in Equation 8. First, the category match function (CMF) value for node J is calculated. If J is a committed node, the CMF value is computed as

$$\rho(\mathbf{w}_J \mid \mathbf{x}) = 1 - \frac{R_J + \max\left\{R_J, \left\|\mathbf{x}-\mathbf{m}_J\right\|_{\mathbf{C}_J}\right\}}{D} \qquad (6)$$

If J is an uncommitted node, its CMF value is constant and is given as

$$\rho(\mathbf{w}_J \mid \mathbf{x}) = 1 \qquad (7)$$

which implies that any pattern will match the shape of the "virtual" category associated with an uncommitted node. Next, the CMF value is compared to the baseline value of the module's *vigilance parameter* $\rho \in [0,1]$ as shown below

$$\rho(\mathbf{w}_J \mid \mathbf{x}) \geq \rho \qquad (8)$$

If the above inequality is not satisfied (J fails the VT), a mismatch reset occurs, during which the CCF value of J is temporarily set to 0, and the competition among F2 layer nodes for **x** continues effectively without the participation of J. Otherwise, we say that pattern **x** chooses node J and now J is eligible to learn pattern **x**. For each input pair (**x**,**y**) a single node J from $\mathrm{ART_a}$ and a single node from $\mathrm{ART_b}$ are chosen. Whether J is permitted to learn **x** depends on the mechanism that regulatesthe association between J and K (see Section III.B).

Let us assume that it has been decided that node J (or K) is permitted to learn a pattern **x**. If J is a committed node its template elements are updated as follows:

$$\mathbf{m}_J^{new} = \mathbf{m}_J^{old} + \frac{\gamma}{2}\left(1 - \frac{\min\left\{R_J^{old}, \left\|\mathbf{x}-\mathbf{m}_J^{old}\right\|_{\mathbf{C}_J^{old}}\right\}}{\left\|\mathbf{x}-\mathbf{m}_J^{old}\right\|_{\mathbf{C}_J^{old}}}\right)(\mathbf{x}-\mathbf{m}_J^{old}) \qquad (9)$$

$$R_J^{new} = R_J^{old} + \frac{\gamma}{2}\left(\max\left\{R_J^{old}, \left\|\mathbf{x}-\mathbf{m}_J^{old}\right\|_{\mathbf{C}_J^{old}}\right\} - R_J^{old}\right) \qquad (10)$$

where $\gamma \in (0,1]$ is the module's *learning parameter*. Fast learning is performed by setting γ=1. Also, the orientation vector $\mathbf{d}_J$ is updated according to Equation 11 only if J has learned a single pattern in the past.

$$\mathbf{d}_J = \frac{\mathbf{x}-\mathbf{m}_J}{\left\|\mathbf{x}-\mathbf{m}_J\right\|_2} \qquad \mathbf{x} \neq \mathbf{m}_J \qquad (11)$$

After the update, $\mathbf{d}_J$ will remain fixed throughout the entire training phase. Finally, if J is an uncommitted node, then the node becomes committed and its template is initialized to $\mathbf{w}_J$=[**x 0** 0]. Let us point out here that similar operations are taking place for F_2 layer nodes in $\mathrm{ART_b}$.

EAM has found major applications in tackling classification tasks. The architecture can be used as a classifier, when the set of class labels is used as its output domain and its $\mathrm{ART_b}$ vigilance parameter is set to ρ=1. From hereon, when we refer to EAM, we will refer to the EAM classifier. Also, when we refer to the vigilance parameter of the EAM classifier, we will mean the corresponding parameter of its $\mathrm{ART_a}$ module. It is worth noting that EAM reduces to some other classifiers with an appropriate choice of network parameters. When training with μ

$=\rho=1$ and then testing with $\mu=1$, $\rho=0$ and $\omega\to\infty$, EAM becomes an L_2-norm 1-Nearest Neighbor classifier [11]. Also, when $\mu=1$ is used for both the training and performance phase, EAM becomes equivalent to HAM. The interested reader may refer to [9] for more properties and characteristics of EA/EAM.

III. BOOSTED ELLISPOID ARTMAP

Due to EAM's design that follows the operating principles of FAM, modifications made in the past to FAM in order to improve some of its shortcomings can be readily and easily applied to EAM. Below we discuss such a modification, which gives rise to Boosted Ellipsoid ARTMAP.

A. THE ROLE OF THE INTER-ART MODULE

EAM features an inter-ART module interconnecting each F_2 layer node j in ART_a with a corresponding node k in ART_b. Each such connection features a weight w_{jk} that is initialized to 0 prior to training. The inter-ART module's role is to keep track of associations between input domain clusters and class labels. As we are going to demonstrate at this point, in EAM only many-to-one mappings are possible. Assume that during training J and K are the chosen nodes in ART_a and ART_b respectively after the presentation of a particular input-output pair $(\mathbf{x},K)$, where K is a class label. If J is uncommitted, then J becomes committed and it will be associated with K by setting $w_{JK}=1$, while for all other weights $w_{Jc}=0$ $c=1..C$. Here, C is the number of class labels. In other words, category J will be labeled as K. Let's assume now that J is already committed and that it is associated to a class label L. If $K=L$, then node J has correctly predicted the class label and is allowed to be updated by pattern $\mathbf{x}$. Otherwise, if $K\neq L$ (wrong prediction), EAM resorts into performing a lateral reset, which will set the CCF value of J to 0 and match tracking will go into effect: the value of the vigilance parameter will be temporarily raised to $\rho(\mathbf{w}_J|\mathbf{x})+\Delta\rho$ where $\Delta\rho$ is some small quantity, and the search for a more suitable category will continue in ART_a. Eventually, the vigilance will be reset to its baseline value after an uncommitted node or a committed node with correct class prediction has been chosen in ART_a. Also, any committed nodes in ART_a that have been reset during the last pattern presentation are reinstated.

B. MODIFICATION TO THE INTER-ART MODULE

The scheme that we have just described adorns EAM with zero resubstitution error in fast off-line learning, since categories in ART_a are always forced to predict the correct label of the training pair. A straightforward approach that improves the generalization qualities of EAM and ameliorates the category proliferation phenomenon is to allow for many-to-many pattern-label associations and regulate the amount of prediction error that each category is permitted to commit. This approach has been successfully implemented in Boosted ARTMAP-S (bARTAMAP-S) [6] and we adopt it in this paper by introducing Boosted Ellipsoid ARTMAP (bEAM).

The new architecture, bEAM, keeps track of how many times categories (nodes) in ART_a are associated to the various class labels in ART_b in a similar fashion to ProbART [12]. However, it features an additional mechanism to control the categories' prediction accuracy. First of all, in bEAM each category remembers the class label of the pattern that initiated its creation. Under the new scheme assume that during training J and K are the chosen nodes in ART_a and ART_b respectively as a response to the presentation of a pair $(\mathbf{x},K)$. If J is uncommitted, then J becomes committed and it will be associated with K by setting $w_{JK}=1$, while for all other weights $w_{Jc}=0$ $c=1..C$. It will also remember the class label of $\mathbf{x}$ that created it by setting its *initial class label* to $I(J)=K$. Considering the alternative case, let us assume that J is already committed with initial class label $I(J)$. If $I(J)=K$, then node J has correctly predicted the class label and is allowed to be updated by pattern $\mathbf{x}$. Furthermore, we set $w_{JK}=w_{JK}+1$. Otherwise, if $I(J)\neq K$, we proceed with the following *prediction test* (PT)

$$\frac{w_{J,I(J)}}{1+\sum_{c=1}^{C} w_{J,c}} \geq 1-\varepsilon \qquad (12)$$

where $\varepsilon\in[0,1]$ is a network parameter of bEAM called *category prediction error tolerance*. If node J passes the PT, then we assume that it did a correct prediction, we set $w_{JK}=w_{JK}+1$ and allow J to be updated by $\mathbf{x}$. In contrast, if J fails the PT, a lateral reset is performed and the match tracking mechanism is activated as in EAM.

The existence of PT in bEAM guarantees that every category in bEAM's ART_a module will not exceed a prediction error of 100ε% with respect to its initial class label. It can be easily shown that for $\varepsilon=0$ bEAM becomes equivalent to EAM, since no category prediction error is allowed. Let us note that bARTMAP-S employs a slightly different PT as shown below

$$\frac{w_{J,D(J)}}{1+\sum_{c=1}^{C} w_{J,c}} \geq 1-\varepsilon \qquad (13)$$

where $D(J)$ is the category's *dominant class label* defined as

$$D(J)=\arg\max_{c=1..C} w_{J,c}=1 \qquad (14)$$

The reason we selected the PT of Equation 12 for bEAM instead of the one in Equation 13 is that it allows bEAM to reduce to an Ellipsoid version of ProbART by setting $\varepsilon=1$. This way any category in ART_a is permitted to predict any class label. For the PT in Equation 13 values of ε less than 0.5 are of uninteresting nature.

The performance phase of bEAM is similar to the one of EAM with only a small exception. If training was performed using some value of $\varepsilon>0$ it is necessary to convert the many-to-many associations of the map field to many-to-one. This is because we usually want the network to predict a single class label for each unlabeled test pattern. Thus, for each category j in ART_a we need to extract its dominant class label $D(j)$ prior to executing bEAM's performance phase. If it happens that $D(j)$ is not unique for some category j (the category strongly predicts more than one class label – a "confused" category), we usually discard it.

First, by tolerating prediction error in the expansion of categories during bEAM training we are able to achieve a non-zero resubstitution error and potentially improve the generalization of bEAM. Secondly, we are able to potentially reduce the number of categories created in ART_a and therefore reduce the complexity of the hypothesis that is learned eventually by bEAM. The latter one is in general true, since for $\varepsilon>0$ patterns corresponding to different class labels are being consolidated into the same cluster eliminating this way the need for the creation of extra categories. As a final, interesting note we mention that, due to its misclassification tolerance feature, bEAM is also capable of coping with inconsistent training patterns, that is, identical patterns associated with different class labels. For an appropriate value of ε bEAM can consolidate inconsistent patterns into a single category. This is not the case with EAM, since inconsistent patterns force the network to become unstable.

IV. PRELIMINARY EXPERIMENTATION

In this section we present some limited, but illustrative experimental results. More specifically our intent is to demonstrate the capability of bEAM as a classifying machine. Towards that end we compare bEAM to EAM and the L_2-norm Restricted Coulomb Energy (RCE) [13] neural network architecture. We have implemented these three architectures (as well as others) as MEX files for use with MathWorks' MATLAB® and they can be found at *http://www.geocities.com/g_anagnostop*. Since all three classifiers feature category representations that are spherical or ellipsoidal, we decided to compare them on the Dichotomized Square problem, which features a linear boundary. A unit-surface square is equally divided into two parts via a perpendicular, dichotomizing boundary. Points on the left side of the boundary are labeled "L" and the remaining "R". We considered two versions of the problem: the Dichotomized Square (DS) problem with no noise (no class overlap) and the Noisy Dichotomized Square (NDS) problem, where each side may contain some patterns belonging to the other side. In order to create a training set for the NDS, we randomly flipped the label of patterns with a 10% probability. Moreover, in our experiments we used a training, cross-validation and test set of cardinalities 200, 5000 and 5000 respectively. All three sets were generated by randomly sampling points from the unit-surface square.

In the sequel, we trained bEAM, EAM and RCE on 30 different orders of training patterns. For each order we selected via cross-validation the best representatives (champions) of each architecture. Finally, we recorded the performance of these champions on a separate test set. For bEAM we used the following training parameter values: ε=0.1, 0.2, ... 0.9 (9 values), μ=0.3, 0.4, ... 1.0 (8 values), ρ=0.85, 0.86, ... 0.98 (14 values), a=0.001, 0.01, 0.1 (3 values), ω=∞ and $D=\sqrt{2}/\mu$ for each value of μ, which totaled 3024 experiments per order. For EAM we used the same parameter settings as bEAM with the exception of ε, which does not apply to EAM. Thus, for EAM we performed 336 experiments per order. Note that fast off-line learning (γ=1) was used for both bEAM and EAM training. For their performance phase we used ρ=0 and ω=∞ to force classification of all patterns. "Confused" categories were removed from bEAM prior to executing its performance phase. Finally, for RCE we used R_{max} = 0.0036, 0.0057, ... 1.2489 (593 values). The results of our experiments are displayed in Figures 2, 3 and Tables I, II. Each point in the figures represents the champion network for a different order of pattern presentation that was selected via cross-validation and then evaluated on the test set.

TABLE I
MISCLASSIFICATION RESULTS FOR THE NOISY DICHOTOMIZED SQUARE

% Misclassification	bEAM	EAM	RCE
Best	***3.02***	5.86	8.5
Average	6.44	7.56	9.86
Worst	9.84	9.3	11.26
Std. Dev.	1.66	0.73	0.66

TABLE II
MISCLASSIFICATION RESULTS FOR THE DICHOTOMIZED SQUARE

% Misclassification	bEAM	EAM	RCE
Best	0.92	***0.66***	0.72
Average	1.24	0.92	1.23
Worst	1.88	1.54	1.94
Std. Dev.	0.23	0.19	0.25

For the NDS problem the overall champion among the architectures is a bEAM network featuring a misclassification error of 3.02% (96.98% correct classification) that utilizes only 20 categories (committed nodes). The next best network is an EAM with 2.84% more error and 16 extra categories. This fact implies that in the presence of noise bEAM may demonstrate better generalization qualities with less complex structure than EAM. It seems that for the NDS problem EAM networks had the tendency of over-fitting the noise, which caused an increase in categories utilized and a decrease in their prediction accuracy. However, they were still able to outperform on average the RCE networks, which seem to have suffered from over-fitting to a higher degree.

Regarding the results for the noiseless DS problem, EAM and RCE networks seem to have met slightly more success in both generalization and data compression than bEAM.

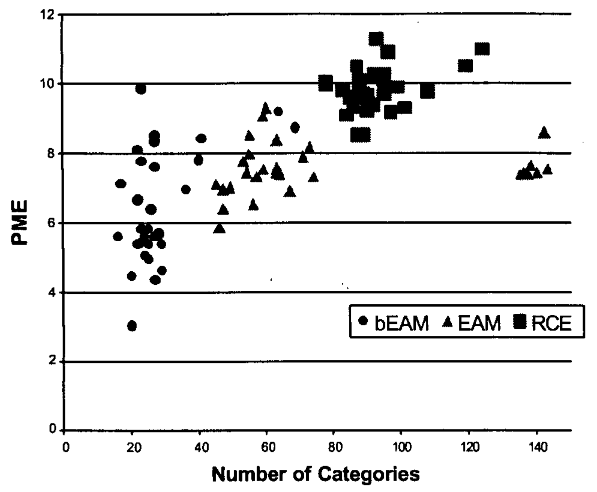

Figure 2: Percent Misclassification Error on the test set versus Number of Categories for the Noisy Dichotomized Square.

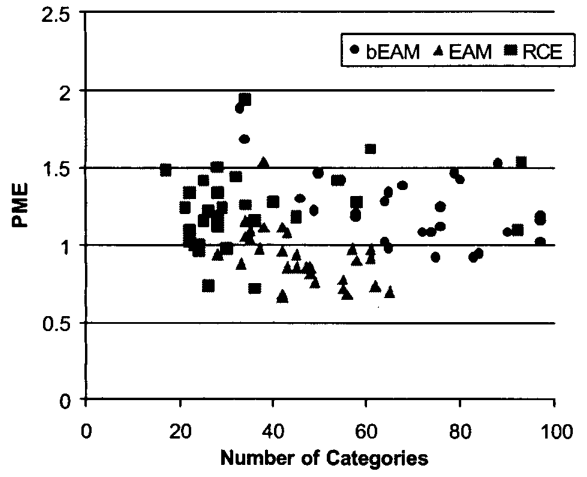

Figure 3: Percent Misclassification Error on the test set versus Number of Categories for the Noisyless Dichotomized Square.

This can be attributed to the fact that in the absence of noise bEAM was less successful to distinguish between the two classes, since it probably created categories encompassing patterns from both classes right on the ideal decision boundary. It was discovered afterwards that the best bEAM network was trained with a value of ε=0.1, which should indicate the lack of noise in the data and that in this case bEAM with ε=0 (i.e., EAM) would exhibit a better performance. In the experiments of this section we used cross-validation to identify good values for bEAM's tolerance ε However, other methods of determining its value a priori are an interesting topic for future research.

V. CONCLUSIONS

In this paper we introduced Boosted Ellipsoid ARTMAP as a variant of Ellipsoid ARTMAP that can potentially reduce its generalization error and ameliorate the phenomenon of category proliferation, while maintaining self-stabilization during fast off-line learning. Furthermore, it is capable of coping with inconsistent training patterns. We also presented some limited, preliminary results that show the potential of Boosted Ellipsoid ARTMAP as a classifier. Finally, by designing this last architecture we have demonstrated that modifications and extensions of Fuzzy ARTMAP can be readily and easily applied to Ellipsoid ARTMAP as well.

REFERENCES

[1] S. Grossberg, "Adaptive Pattern Recognition and Universal Encoding II: Feedback, Expectation, Olfaction, and Illusions", Biological Cybernetics, Vol. 23, pp. 187-202, 1976.

[2] G.C. Anagnostopoulos and M. Georgiopoulos, "Ellipsoid ART and ARTMAP for Incremental Unsupervised and Supervised Learning", Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '01), Vol. 2, pp. 1221-1226, Washington, Washington D.C., July, 2001.

[3] G.C. Anagnostopoulos and M. Georgiopoulos, "Hypersphere ART and ARTMAP for Unsupervised and Supervised Incremental Learning", Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00), Vol. 6, pp. 59-64, Como, Italy, 2000.

[4] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds and D.B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps", IEEE Transaction on Neural Networks, Vol. 3:5, pp. 698-713, 1992.

[5] J.C. Bezdek,, T. Reichherzer, G.S. Lim, and Y. Attikiouzel, "Multiple-prototype Classifier Design", IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 28:1, pp. 67-79, 1998.

[6] S.J. Verzi, G.L. Heileman, M. Georgiopoulos and M. J. Healy, "Rademacher Penalization Applied to Fuzzy ARTMAP and Boosted ARTMAP", Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '01), Vol. 2, pp. 1191-1196, Washington, Washington D.C., July, 2001.

[7] V.N. Vapnik, The Nature of St atistical Learning Theory, Springer Verlag, New York, 1995.

[8] V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.

[9] G.C. Anagnostopoulos, Novel Approaches in Adaptive Resonance Theory for Machine Learning, Doctoral Dissertation, University of Central Florida, Orlando, Florida, USA, 2001.

[10] G.A. Carpenter, S. Grossberg and D.B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System", Neural Networks, Vol. 4:6, pp. 759-771, 1991.

[11] R.O. Duda, P.E.Hart and D.G. Stork, Pattern Classification, 2nd Ed., Wiley & Sons, New York, 2001.

[12] S. Mariot and R.F. Harrison, "A Modified Fuzzy ARTMAP Architecture for the Approximation of Noisy Mappings", Neural Networks, Vol. 8:4, pp. 619-641, 1995.

[13] M.H. Hassoun, Fundamentals of Artificial Neural Networks, MIT Press, Cambridge, Massachusetts, 1995.