Boosted Ellipsoid ARTMAP

Georgios C. Anagnostopoulos[*a], Michael Georgiopoulos[**a], Steven J. Verzi[***b]
& Gregory L. Heileman[****b]
[a]School of EE & CS/University of Central Florida
[b]Department of EE & CpE/University of New Mexico

ABSTRACT

Ellipsoid ARTMAP (EAM) is an adaptive-resonance-theory neural network architecture that is capable of successfully performing classification tasks using incremental learning. EAM achieves its task by summarizing labeled input data via hyper-ellipsoidal structures (categories). A major property of EAM, when using off-line fast learning, is that it perfectly learns its training set after training has completed. Depending on the classification problems at hand, this fact implies that off-line EAM training may potentially suffer from over-fitting. For such problems we present an enhancement to the basic Ellipsoid ARTMAP architecture, namely Boosted Ellipsoid ARTMAP (bEAM), that is designed to simultaneously improve the generalization properties and reduce the number of created categories for EAM's off-line fast learning. This is being accomplished by forcing EAM to be tolerant about occasional misclassification errors during fast learning. An additional advantage provided by bEAM's desing is the capability of learning inconsistent cases, that is, learning identical patterns with contradicting class labels. After we present the theory behind bEAM's enhancements, we provide some preliminary experimental results, which compare the new variant to the original EAM network, Probabilistic EAM and three different variants of the Restricted Coulomb Energy neural network on the square-in-a-square classification problem.

Keywords: adaptive resonance theory, Ellipsoid ARTMAP, classification

1. 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[2] appeared as a generalization of Hyper-sphere ARTMAP[3] (HAM). It is an alternative architecture to Fuzzy ARTMAP[4] (FAM) 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 suitably 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-

[*] anagnostop@email.com; phone 1 407 228-1231; http://www.geocities.com/g_anagnostop; 4132 Lake Underhill Road #303, Orlando, FL USA 32803; [**] michaelg@mail.ucf.edu; phone 1 407 823-5338; fax 1 407 823-5835; School of EECS, University of Central Florida, 4000 Central Florida Blvd., Orlando, FL USA 32816-2450; verzi@eece.unm.edu; phone : 1 505 245-9970; fax 1 505 277-1413; 12911 Kachina Pl. NE Apt. B, Albuquerque, NM USA 87131; heileman@eece.unm.edu; phone 1 505 277-4011; fax 1 505 277-1413; EECE Department University of New Mexico Albuquerque, NM USA 87131.

Applications and Science of Computational Intelligence V, Kevin L. Priddy, Paul E. Keller, Peter J. Angeline, Editors, Proceedings of SPIE Vol. 4739 (2002) © 2002 SPIE · 0277-786X/02/$15.00

stabilization in off-line mode via *fast learning* (see Section 2.3). 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 learning has completed, EAM has memorized its training set to perfection without committing any classification 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 problem setting 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[6] (bARTMAP-S). It is a variant of FAM with a tunable, universal misclassification tolerance parameter that indirectly influences the level of the resubstitution error. It has been shown[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 this paper we adopt a similar approach to bARTMAP-S, which we apply 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. Let us emphasize at this point that bARTMAP-S's training principles can be directly applied to EAM by virtue of the design similarities of the latter to FAM. We call the new architecture Boosted Ellipsoid ARTMAP (bEAM) to indicate its connection to the principals of bARTMAP-S. As we will show later in Section 3.2, a byproduct of bEAM's enhanced training procedure is the ability of coping with contradictory training examples. The rest of the paper is organized as follows: 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 merit of our approach.

2. ELLIPSOID ARTMAP

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

2.1 Brief overview of the Ellipsoid ARTMAP architecture

A block diagram of EAM is depicted in Figure 1. As shown, 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^2 (EA) 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 $\mathbf{w}_{jk}$ 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 their input patterns. Both F_1 and F_2 layers consist of an array of nodes that are interconnected across the 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.

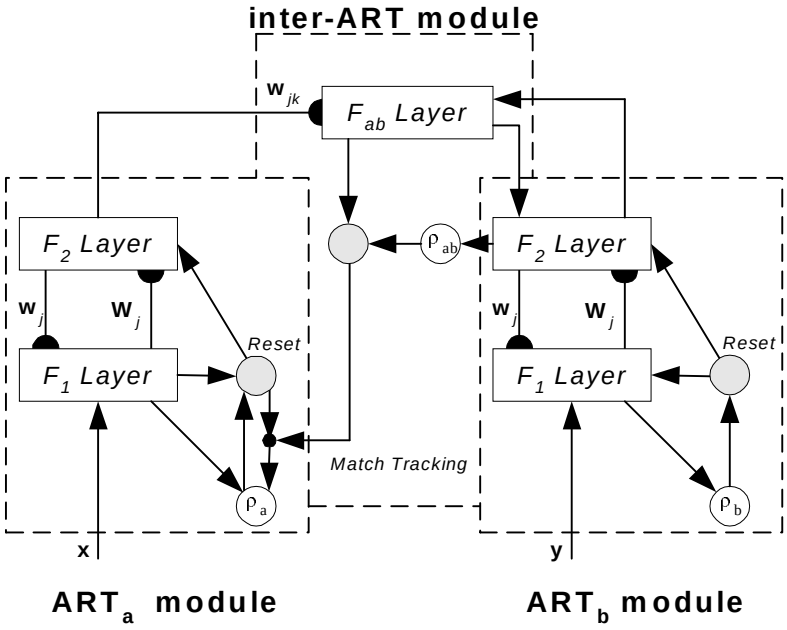

Figure 1: Block diagram of Ellipsoid ARTMAP.

2.2 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 2, 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 2 is called the *representation region* of category *j* and it encompasses all patterns that the category has encoded.

During the training phase of 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 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 has been shown[9] that exactly this behavior guarantees the fast off-line, self-stabilizing learning property of EA and EAM.

Once it is has been decided that a pattern $\mathbf{x}$ 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[10] (FA) 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 $\mathbf{x}$. Similarly, in EA/EAM, Hypersphere-ART[3] (HA) 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.

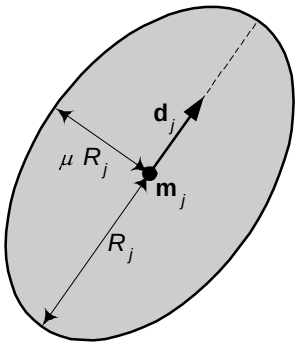

Figure 2: Ellipsoid ART category embedded in a 2-dimensional input space.

2.3 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 ART_a and 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 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 (Weber Law) CCF value is defined as

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

where

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

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. For the Weber Law CCF in Equation 1, $a>0$ is the *choice parameter*; the network parameter $D>0$ is usually set as

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

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

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

where the parameter ω is chosen as $\omega \geq 0.5$ to ensure stable learning in EAM. The node *J* featuring the maximum CCF value and smallest index *j* is considered to be the winner of the winner-take-all competition.

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

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

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

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

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

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

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

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 until the presentation of the next training pattern 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 ART_a and a single node *K* from ART_b are chosen. Whether *J* (*K*) is permitted to learn **x** (**y**) depends on the mechanism that regulates the association between *J* and *K* (see Section 3.2).

Let us assume it has been decided that node *J* (or *K*) is permitted to learn a pattern **x**. If *J* is an uncommitted node, a new EA category will be created by committing *J* and initializing its template elements to

$$\begin{aligned} \mathbf{m}_J &= \mathbf{x} \\ \mathbf{d}_J &= \mathbf{0} \\ R_J &= 0 \end{aligned} \tag{9}$$

A category *j* featuring $R_j = 0$ is called a *point category*. On the other hand, if *J* is a committed node its template elements are updated as follows:

$$\mathbf{m}_J^{new} = \mathbf{m}_J^{old} + \frac{\gamma}{2}\left(1 - \frac{\min\left\{R_J^{old}, \left\|\mathbf{x} - \mathbf{m}_J^{old}\right\|_{C_J^{old}}\right\}}{\left\|\mathbf{x} - \mathbf{m}_J^{old}\right\|_{C_J^{old}}}\right)\left(\mathbf{x} - \mathbf{m}_j^{old}\right) \tag{10}$$

$$R_J^{new} = R_J^{old} + \frac{\gamma}{2}\left(\max\left\{R_J^{old}, \left\|\mathbf{x} - \mathbf{m}_J^{old}\right\|_{C_J^{old}}\right\} - R_J^{old}\right) \tag{11}$$

where $\gamma \in (0,1]$ is the module's *learning parameter*. Fast learning is performed by setting $\gamma = 1$. Also, the orientation vector $\mathbf{d}_J$ is updated according to Equation 12 only if *J* was a point category ($R_J = 0$) prior to the update.

$$\mathbf{d}_J = \frac{\mathbf{x} - \mathbf{m}_J}{\|\mathbf{x} - \mathbf{m}_J\|_2} \qquad \mathbf{x} \neq \mathbf{m}_J \tag{12}$$

After the update is performed, $\mathbf{d}_J$ will remain fixed throughout the entire training phase. Let us point out here that similar operations are taking place for F_2 layer nodes in ART_b.

EAM has found major applications in tackling classification problems. The architecture can be used as a classifier, when the set of class labels is used as its output domain and its ART_b vigilance parameter is set to $\rho = 1$. From hereon, when we refer to EAM, we will refer to the EAM classifier. Also, when we refer to the network parameter of the EAM classifier, we will mean the corresponding parameters of its ART_a module. It is worth noting that EAM reduces to some other classifiers with an appropriate choice of network parameters. When training with $\mu = \rho = 1$ and then testing with $\mu = 1$, $\rho = 0$ and $\omega \to \infty$, EAM becomes an L_2-norm 1-Nearest Neighbor classifier[11]. In general, when $\mu = 1$ is used for both

the training and performance phase, EAM becomes equivalent to HAM. The interested reader may refer to the references[9] for more properties and characteristics of EA/EAM.

3. 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 Ellispoid ARTMAP.

3.1 The role of the map field

EAM features an inter-ART module (the map field) interconnecting each F_2 layer node j in ART_a with a corresponding node k in ART_b. Each such connection features a weight $\mathbf{w}_{jk}$ that is initialized to 0 prior to training. The map field'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 $\mathbf{w}_{JK}=1$, while for all other weights $\mathbf{w}_{Jc}=0$ $c=1..C$. 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 until the next training pattern is presented 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.

3.2 Boosted Ellipsoid ARTMAP

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 and ameliorates the category proliferation phenomenon in EAM 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[6] (bARTAMAP-S) and we adopt it in this paper by introducing Boosted Ellipsoid ARTMAP (bEAM).

The new architecture, bEAM, keeps track of the frequency with which 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 $\mathbf{w}_{JK}=1$, while for all other weights $\mathbf{w}_{Jc}=0$ $c=1..C$. 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 $\mathbf{w}_{JK}=\mathbf{w}_{JK}+1$. Otherwise, if $I(J)\neq K$, we proceed with the following *prediction test* (PT)

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

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 $\mathbf{w}_{JK}=\mathbf{w}_{JK}+1$ 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 standard EAM training.

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 ε=0 bEAM becomes equivalent to EAM, since no category prediction error is allowed. Let us note here that bARTMAP-S employs a slightly different PT as shown below

$$\frac{w_{J,D(J)}}{1+\sum_{c=1}^{C} w_{J,c}} \geq 1-\varepsilon \tag{14}$$

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

$$D(J) = \arg\max_{c=1..C} w_{J,c} = 1 \tag{15}$$

The reason we selected the PT of Equation 13 for bEAM instead of the one in Equation 14 is that it allows bEAM to reduce to an Ellipsoid version of ProbART, namely *Probabilistic EAM* (ProbEAM), by setting ε=1. This way any category in ART_a is permitted to predict any class label. Note that for the PT in Equation 14 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 ε>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. Pseudocode for the bEAM classifier is provided in the Appendix.

First, by tolerating prediction error in the expansion of categories during bEAM training we are guaranteed to achieve a non-zero resubstitution error and potentially improve the generalization of bEAM. Secondly, we are potentially able to 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 ε >0 patterns corresponding to different class labels will be 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 contradicting 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. It can be shown that if a pattern $\mathbf{x}$ occurs L times in the training set and each time it is associated with a different class label, then bEAM will be able to cope with this contradictory evidence, when we choose

$$\varepsilon \geq 1-\frac{1}{L} \tag{16}$$

4. EXPERIMENTAL RESULTS

In this section we present some limited, but illustrative experimental results about bEAM's behavior and classification performance. Towards that end we compare bEAM to the L_1-norm, L_2-norm and L_∞-norm Restricted Coulomb Energy[13] (RCE) neural network architecture. Also, we consider EAM and ProbEAM as special cases of bEAM. We have implemented the aforementioned 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*. We compare the classification performance of these architectures on the *noisy square-in-the-square* (NSIS) problem. This particular classification task considers a square of surface 1/2 centered inside the unit square. Patterns inside the inner square are labeled '1', while the ones outside it are labeled '0'. Noise is introduced into the classification problem by flipping the class label of patterns from '0' to '1' and vice versa with a probability of 0.15. Obviously the optimal classifier for this task would achieve on average 85 percent correct classification (PCC) performance on any test set. In our experiments we used training, cross-validation and test sets of cardinalities 100, 1000 and 5000 respectively. All three sets were generated by randomly sampling points from the unit-surface square and then introducing noise as we have previously described.

In the sequel, we trained bEAM (including standard EAM and ProbEAM as special cases) and the three version of the RCE network on 100 different orders of training patterns. For each order we selected via cross-validation the best (in terms of PCC) 100 representatives for each architecture. Finally, we recorded the performance of these elite networks on a separate test set. For bEAM we used the following training parameter values: ε =0.0, 0.05, ... 1.0 (21 values), μ =0.2, 0.4, … 1.0 (5 values), ρ =0.0, 0.02, … 0.98 (50 values), a=0.001 (1 value), γ =1 (fast learning), ω =∞ and $D = \sqrt{2} / \mu$ for each value of μ, which totaled 525000 bEAM off-line training sessions. Each training phase of bEAM was run until the corresponding network stabilized. For bEAM's 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. As we have mentioned in Section 3.2, bEAM becomes equivalent to EAM for ε =0.0 and equivalent to ProbEAM for ε =1.0. Finally, for the RCE networks we used R_{max} = 0.0036, 0.0057, … 1.2489 (593 values) and performed a total of 177900 off-line training sessions. The results of our experiments are displayed in Figure 3 and Tables I.

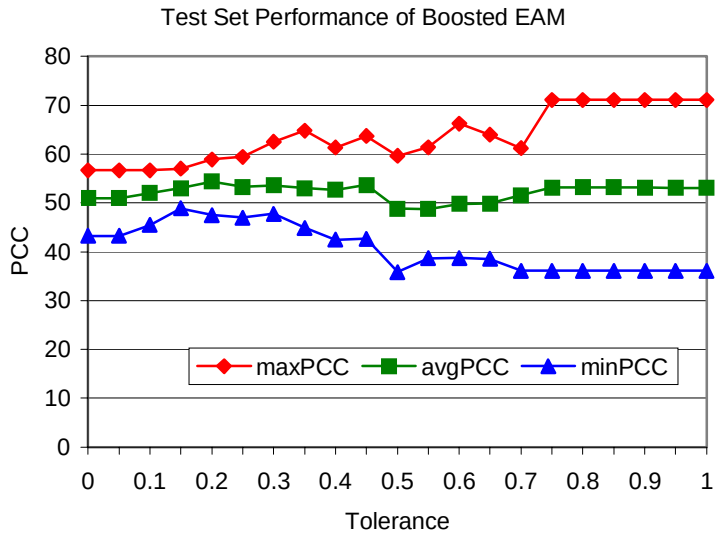

Figure 3: Percent Correct Classification vs. values of ε on the test set of the best 100 Boosted EAM networks for the noisy square-in-the-square problem.

Figure 3 displays the maximum, average and minimum PCC on the test set of the 100 best bEAM networks, which were determined on the basis of their classification performance on the cross-validation set. The PCC statistics are graphed versus the value of the misclassification tolerance parameter ε. We observe that as ε is increases (as bEAM transitions from standard EAM to ProbEAM behavior), the maximum PCC has the tendency to increase as well. Simultaneously, however, the variability in PCC also increases, since, in general, the minimum PCC has the tendency to decrease. For values of ε>0.7 the best bEAM networks feature a PCC that is 15% more than the one of the best EAM architecture. The previous fact demonstrates the efficiency of bEAM to cope with data-inherent noise through better generalization properties than EAM in the NSIS problem. Evidently, for this problem EAM tends to over-fit the training data more than bEAM does. A final observation is that for values of ε >0.8, the tolerance ε has no effect on the training of bEAM for the classification problem at hand: choosing ε =0.85 or ε =1.0 (ProbEAM) seems to yield identical architectures for the same values of the rest of the network parameters and the same order of training pattern presentation.

Table 1 compares the performance (PCC on the test set) statistics of bEAM to the ones of EAM, ProbEAM, and the three RCE variants. Our first observation is that the family of ellipsoid networks outperforms the RCE family by at least 2% in maximum PCC. The RCE networks seem to suffer from over-fitting like the standard EAMs do. The best 100 networks seem to be bEAMs that were trained with ε =0.75. Although these networks feature a maximum PCC value equal to the one exhibited by the 100 best ProbEAMs, their average PCC value is almost 1.5% better than the

corresponding value for the ProbEAMs. This difference in average PCC is significant, since 5000 test patterns were used to evaluate performances.

Table 1: Comparison of PCC statistics on the test set for the champion network of each type.

Architecture	Maximum PCC	Average PCC	Minimum PCC	Std. Dev. PCC
Boosted EAM	71.12	53.1992	36.12	5.700601
Probabilistic EAM	71.12	53.0564	36.12	5.735463
EAM	56.66	50.9618	43.26	2.943694
L_1-norm RCE	52.3	50.761	49.66	0.485502
L_2-norm RCE	52.66	50.6864	48.52	1.018547
L_∞-norm RCE	53.28	51.0718	49.62	0.926552

5. SUMMARY

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, the newly introduced variant 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, **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 Statistical Learning Theory, Springer Verlag, New York, 1995.
8. V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.
9. 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.
10. R.O. Duda, P.E.Hart and D.G. Stork, Pattern Classification, 2nd Ed., Wiley & Sons, New York, 2001.
11. G.C. Anagnostopoulos, Novel Approaches in Adaptive Resonance Theory for Machine Learning, Doctoral Dissertation, University of Central Florida, Orlando, Florida, USA, 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.

APPENDIX

The following pseudocode implements the performance phase and the training phase for off-line, fast learning (γ=1) of the Boosted Ellipsoid ARTMAP classifier. Lines starting with "//" imply comments and the assignment operator is denoted as ":=".

// Training Phase of Boosted EAM:
// Set in_training_phase:=*true*
// Provide values for the network parameters: $\varepsilon \in [0,1]$, $\mu \in (0,1]$, $\rho \in [0,1]$, $\Delta\rho << 1, a > 0$, $\omega \geq 0.5$
// To use bEAM as ProbEAM set ε=1
// To use bEAM as EAM set ε=0
// To use bEAM as HAM set ε=0 and μ=1
// Set the initial number of committed nodes N:=0
// It is assumed that the training set consists of P>1 normalized pattern-label pairs
// of the form $\{\mathbf{x}_p, l_p\}$ p=1..P, where $l_p \in \{1,..,C\}$ and $\mathbf{x}_p \in [0,1]^M$ (M is the dimensionality
// of the patterns).
// Set all the map field weights wjk:=0 j=1..N_{un} and k=1..C,
// where N_{un} is the total number of uncommitted F_2-layer nodes in the ART_a module
// of bEAM and C is the number of class labels. Note that prior to training bEAM
// starts off with all F_2-layer nodes in the ART_a module being uncommitted.
//
// Performance Phase of Boosted EAM:
// Set in_training_phase:=*false*
// Provide values for the network parameters: $\rho \in [0,1]$, $a > 0$, $\omega \geq 0.5$
// It is assumed that during training N categories have been created and their description
// is summarized in the pairs $\{\mathbf{w}_j, D(j)\}$ j=1..N, where $D(j)$ is the dominant label of category j.
// It is assumed that the test set consists of P>1 normalized, unlabeled patterns $\mathbf{x}_p \in [0,1]^M$ p=1..P
// To force classification of all test patterns use ρ=0, $\omega \to \infty$ and eliminate all categories j,
// for which $D(j)$=*confused*.

// Initialization

Set $D := \dfrac{\sqrt{M}}{\mu}$, $T_u := \dfrac{D}{2D\omega + a}$

DO
{ // Pattern List Presentation Loop.
FOR p:=1..P DO
{ Set S:=∅

// Loop for the calculation of CMF and CCF values.
FOR j:=1..N DO

{ Calculate $dis(\mathbf{x}_p, \mathbf{w}_j) := \max\left\{0, \left\|\mathbf{x}_p - \mathbf{m}_j\right\|_{\mathbf{C}_j} - R_j\right\}$ with help of Equation 2

IF $dis(\mathbf{x}_p, \mathbf{w}_j) > 0$ THEN

{ Calculate $\rho(\mathbf{w}_j \mid \mathbf{x}_p) := 1 - \dfrac{2R_j + dis(\mathbf{x}_p, \mathbf{w}_j)}{D}$

// Perform the Vigilance Test (VT)
IF $\rho(\mathbf{w}_j \mid \mathbf{x}_p) < \rho$ THEN NEXT j

}

// Observe that, if $dis(\mathbf{x}_p, \mathbf{w}_j) = 0$, category *j* is guaranteed to pass the VT.

Calculate $T(\mathbf{w}_j \mid \mathbf{x}_p) = \dfrac{D - 2R_j - dis(\mathbf{x}_p, \mathbf{w}_j)}{D - 2R_j + a}$

Include *j* in *S*.

} // End of *j*-loop

Set *J*:=*none*

WHILE *S*≠∅ DO
{ // Find the node *J* with maximum CCF value.
// If more than one node features the maximum CCF value,
// pick the one of lowest index.
Find $J := \inf \arg\max_{j \in S} \{T(\mathbf{w}_j \mid \mathbf{x}_p)\}$

IF in_training_phase=*false* THEN
{ // Perform the Commitment Test (CT).
IF $T(\mathbf{w}_J \mid \mathbf{x}_p) < T_u$ THEN set *J*:=*none*
EXIT WHILE LOOP
}

// The following actions are performed only during the training phase.
// Perform the Commitment Test (CT).
IF $T(\mathbf{w}_J \mid \mathbf{x}_p) < T_u$ THEN exclude *J* from *S*.

ELSE IF $T(\mathbf{w}_J \mid \mathbf{x}_p) \geq T_u$ THEN
{ // Category *J* passed the CT.
// Now perform the Prediction Test (PT).

IF l_p=*D*(*J*) OR $\dfrac{w_{J,I(J)}}{1 + \sum_{c=1}^{C} w_{J,c}} \geq 1 - \varepsilon$ THEN EXIT WHILE LOOP

ELSE IF IF $l_p \neq D(J)$ AND $\dfrac{w_{J,I(J)}}{1 + \sum_{c=1}^{C} w_{J,c}} < 1 - \varepsilon$ THEN
{ // Category *J* is not suitable for update,
// since it failed the PT. We must search for
// another eligible category.

Exclude *J* from *S*.

// Perform Match Tracking (MT)
Set $\rho := \rho(\mathbf{w}_J \mid \mathbf{x}_p) + \Delta\rho$

Set *J*:=*none*
}
}
} // End of while-loop

IF *J*=*none* THEN
{ // An uncommitted node was chosen by $\mathbf{x}_p$.
// This means that $\mathbf{x}_p$. is an atypical pattern that could not be classified.

IF in_training_phase=*false* THEN report that pattern $\mathbf{x}_p$'s label is *unknown*.
ELSE IF in_training_phase=*true* THEN
{ // Create new category by committing node *J*
Set $\mathbf{m}_J := \mathbf{x}_p, \mathbf{d}_J := \mathbf{0}, R_J := 0, I(J) := l_p, D(J) := l_p$
Set *N*:=*N*+1
}
}
ELSE IF *J*≠*none* THEN
{ // An already committed node was chosen by $\mathbf{x}_p$.

IF in_training_phase=*false* THEN report that pattern $\mathbf{x}_p$'s label is *D*(*J*).
ELSE IF in_training_phase=*true* THEN
{ // Category *J* is eligible to be updated.
Set $w_{J,l_p} := w_{J,l_p} + 1$

// Recalculate the dominant label of *J*
Find $L := \arg\max_c \{w_{J,c}\}$
IF *L* is a non-unique maximizer of $w_{J,c}$ THEN set *D*(*J*):=*confused*
ELSE IF *L* is the unique maximizer of $w_{J,c}$ THEN set *D*(*J*):=*L*

// Update category *J*.
// Observe that due to bEAM's learning law in Equations
// 10-12, if $dis(\mathbf{x}_p, \mathbf{w}_J) = 0$, no update occurs.
IF $dis(\mathbf{x}_p, \mathbf{w}_J) > 0$ THEN

{ IF R_J=0 THEN set $\mathbf{d}_J := \dfrac{\mathbf{x}_p - \mathbf{m}_J}{\left\|\mathbf{x}_p - \mathbf{m}_J\right\|_2}$

Calculate $\Delta R_J := \frac{1}{2} dis(\mathbf{x}_p, \mathbf{w}_J)$
Set $R_J := R_J + \Delta R_J$
Set $\mathbf{m}_J := \mathbf{m}_J + \Delta R_J \dfrac{\mathbf{x}_p - \mathbf{m}_J}{\left\|\mathbf{x}_p - \mathbf{m}_J\right\|_{C_J}}$
}
}
}

} // End of *p*-loop

// The outer while-loop is only executed once during performance phase
IF in_training_phase=*false* THEN EXIT DO-WHILE

} WHILE no new categories have been created and no categories have been updated during the last list presentation (epoch).