Experiments with μARTMAP: Effect of the Network Parameters on the Network Performance

M. Zhong (*), B. Rosander (*), M. Georgiopoulos (*), G. Anagnostopoulos (**),
M. Mollaghasemi (*), and S. Richie (*)

(*) University of Central Florida, Orlando, FL
(**) Florida Institute of Technology, Melbourne, FL

Abstract

Fuzzy ARTMAP (FAM) is currently considered to be one of the premier neural network architectures in solving classification problems. One of the limitations of Fuzzy ARTMAP that has been extensively reported in the literature is the category proliferation problem. That is Fuzzy ARTMAP has the tendency of increasing its network size, as it is confronted with more and more data, especially if the data are of the noisy and/or overlapping nature. To remedy this problem a number of researchers have designed modifications to the training phase of Fuzzy ARTMAP that had the beneficial effect of reducing this category proliferation. One of these modified Fuzzy ARTMAP architectures was the one proposed by Gomez-Sanchez, and his colleagues, referred to as μARTMAP. In this paper we present reasonable analytical arguments that demonstrate of how we should choose the range of the some of the μARTMAP other parameters. Furthermore, we perform an exhaustive experimentation to find the best μARTMAP network for a variety of problems (simulated and real problems). Through this experimentation we were able to identify good default values for the μARTMAP network parameters in a variety of problems. Finally, we identified the best performing μARTMAP network (from the set of parameter values that we have experimented with) and we compared it with other ART networks, including other ART networks that claim that resolve the category proliferation problem in Fuzzy ARTMAP.

1. Introduction

The Adaptive Resonance Theory (ART) was developed by Grossberg (see Grossberg, 1976). One of the most celebrated ART architectures is Fuzzy ARTMAP (see Carpenter, et al., 1992), which has been successfully used in the literature for solving a variety of classification problems. Some of the advantages that Fuzzy ARTMAP possesses is that it can solve arbitrarily complex classification problems, it converges quickly to a solution (within a few presentations of the list of the input/output patterns belonging to the training set), it has the ability to recognize novelty in the input patterns presented to it, it can operate in an on-line fashion (new input/output patterns can be learned by the system without re-training with the old input/output patterns), and it produces answers that can be explained with relative ease.

One of the limitations of Fuzzy ARTMAP that has been extensively reported in the literature is the category proliferation problem. That is Fuzzy ARTMAP has the tendency of increasing its network size, as it is confronted with more and more data, especially if the data are of the noisy and/or overlapping nature. To remedy this problem a number of researchers have designed modifications of the training phase of Fuzzy ARTMAP that have had the beneficial effect of reducing this category proliferation (e.g., Verzi, et al., 2001, Anagnostopoulos, et al., 2003, and Gomez-Sanchez, et al. 2002).

In this paper we focus our attention on one of these Fuzzy ARTMAP modifications that is the one introduced by Safe μARTMAP (see Gomez-Sanchez, et al., 2001). As it has been reported in the literature, μARTMAP's approach to reduce the category proliferation problem is to allow categories in Fuzzy ARTMAP to encode input patterns that belong to different labels, thus eliminating the need of creating a new category every time an input pattern appeared in the vicinity of categories of different labeling than the one that the input pattern possessed. Furthermore, μARTMAP allowed some of the mixed label categories to be destroyed if they were too entropic (i.e., mixing of the labels within a category was too excessive). μARTMAP enforces the category destruction through a set of maximum allowed entropic thresholds. After a category is destroyed μARTMAP allows the creation of categories of smaller size, than the category that is destroyed. The performance (size of architecture created, and classification accuracy achieved on unseen data) by μARTMAP depends on the choice of the network parameters (i.e., entropic thresholds, baseline vigilance parameter, choice parameter, and order of pattern presentation in the training set). As an enhanced version of μARTMAP, Safe μARTMAP does not allow a category to expand too quickly, and avoids the creation of highly overlapped categories.

In this paper we contribute to the existing μARTMAP literature and in a bigger context the ART literature by presenting reasonable analytical arguments that demonstrate of how we could choose the range of the entropic thresholds. Furthermore, we perform an exhaustive experimentation to find the best μARTMAP network for a variety of problems (simulated data and real data). Through this experimentation we were able to define good default values for the μARTMAP network parameters, applicable to a variety of problems. Finally, we identified the best performing μARTMAP network (from the set of network parameter values that we have experimented with) and we compared it with other best performing ART networks, such as Fuzzy ARTMAP (see Carpenter, et al., 1992), Ellipsoidal ARTMAP (see Anagnostopoulos, et al., 2001), Gaussian ARTMAP (see Williamson, 1996 and 1997), and their semi-supervised versions (see Anagnostopoulos, et al., 2003).

2. The μARTMAP Architecture

The block-diagram of the μARTMAP architecture is shown in Figure 1. The μARTMAP architecture of the block diagram of Figure 1 has three major layers. The input layer (F_1^a) where the input patterns (designated by $\mathbf{I}$) are presented, the category representation layer (F_2^a) where compressed representations of these input patterns are formed (designated as $\mathbf{w}_j^a$ and called templates), and the output layer (F_2^b) that holds the labels of the categories formed in the category representation layer. Another layer shown in Figure 1, and designated by F_0^a , is a pre-processing layer and its functionality is to pre-process the input patterns prior to their presentation to μARTMAP. The pre-processing operation, called complementary coding, takes an input pattern $\mathbf{a}$ and expands it by appending to it the complement of a, designated as $\mathbf{a}^c$. That is, the input pattern $\mathbf{I}$ to μARTMAP is now equal to

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c)$$

where $\mathbf{a}^c = \mathbf{1} - \mathbf{a}$, or in other words every component of the complement vector is equal 1 minus the corresponding component of the original vector. It is assumed here that every component of the original vectors $\mathbf{a}$ lies in the interval $[e, 1-e]$ (where e is a small positive number, as explained in "μARTMAP Parameters"), and if it does not we normalize it so that it does. The number of nodes in the input layer of μARTMAP is equal to $2M_a$ (where M_a is the dimensionality of the vector $\mathbf{a}$), and we use the index *i* to designated a generic node in the input layer. The number of committed nodes in the category representation layer is equal to N_a (and this number is changing dynamically throughout the training process of μARTMAP as more nodes are committed to correctly encode the input patterns) and we use the generic index *j* to designate one of these nodes. The number of nodes in the output layer of μARTMAP is equal to N_b (and N_b corresponds to the number of distinct labels of the pattern classification task that μARTMAP is focusing on), and we use the generic index *k* to designate a node in the output layer.

There are a number of weights in the μARTMAP architecture that are worth mentioning: (a) The vector of weights (templates) emanating from every node in the category representation layer and converging to all the nodes in the input layer. For example, the vector of weights emanating from node *j* in the category representation layer and converging to all the nodes in the input layer is designated by $\mathbf{w}_j^a = (w_{j1}^a, \ldots, w_{ji}^a, \ldots, w_{j2M_a}^a)$ and represents the compressed features of all the input patterns that chose node *j* as their representative node in the training process of μARTMAP. (b) The vector of weights emanating from every node in the category representation layer and converging to all the nodes in the output layer. For example, the vector of weights emanating from node *j* in the category representation layer and converging to all the nodes in the output layer is designated by $\mathbf{W}_j^{ab} = (W_{j1}^{ab}, \ldots, W_{jk}^{ab}, \ldots, W_{jN_b}^{ab})$, and component W_{jk}^{ab} of this weight vector represents the number of times that node *j* has been chosen by an input pattern, it encoded this input pattern, and the label of this input pattern was label *k*.

μARTMAP operates in two phases: The training phase and the performance phase. In the training phase of μARTMAP we have a collection of input/associated labels pairs (called training set), and we present it to μARTMAP, one input/associated label pair at a time, in a manner that will be further explained below, until μARTMAP maps the input patterns to their associated labels within a certain degree of tolerance. This tolerance is explicitly expressed by the entropic thresholds that μARTMAP enforces. A generic input/output pair in the training set is designated as $\{\mathbf{I}, label(\mathbf{I})\}$, and the labels of patterns are represented by an index k, where $1 \le k \le N_b$. The performance phase of μARTMAP will be explained after the training phase has been discussed.

The training phase of μARTMAP is succinctly described as follows (Steps 1-2):

1. (Learning Phase) Find the nearest category in the category representation layer of μARTMAP that resonates with the input patterns.
    a. If the label of the input pattern is such that the entropy of this category does not exceed a pre-defined threshold update the weights of this category.
    b. Otherwise, reset the winner, and try the next winner. Uncommitted nodes are chosen if and only if we cannot find a winner node from the list of already committed nodes.
2. (Offline Evaluation Phase) After the learning phase is finished (i.e., all input/associated label pairs of the training set have chose a committed node) we present all the patterns again to check the total entropy of the created categories, without changing any $\mathbf{w}_j^a$ vector. One pass of the learning phase and the offline evaluation phase is called one epoch.
    a. If the total entropy is below a designated threshold, training is completed.
    b. If not, the category that contributes the most to the total entropy value is destroyed, the vigilance threshold in μARTMAP is increased to be slightly higher than the vigilance level of the category destroyed, and the next epoch will be started. In the learning phase of the next epoch, however, we present to μARTMAP only the training patterns that chose the destroyed category in the learning phase (rather than offline evaluation phase) of this epoch or the previous epochs. In the offline evaluation of the next epoch, we still present all the patterns.

The nearest category (mentioned in Step 1) to an input pattern $\mathbf{I}$ presented to μARTMAP is determined by finding the category that maximizes the function:

$$T_j^a(\mathbf{I}, \mathbf{w}_j^a, \alpha) = \frac{|\mathbf{I} \wedge \mathbf{w}_j^a|}{\alpha + |\mathbf{w}_j^a|} \tag{1}$$

The above function is called the bottom-up input (or choice function) pertaining to the F_2^a layer node j with category representation (template) equal to the vector $\mathbf{w}_j^a$, due to the presentation of input pattern $\mathbf{I}$. This function obviously depends on the μARTMAP parameter α, called choice parameter, which assumes values in the interval $(0, \infty)$. In most simulations of ART architectures the useful range of α is the interval (0, 10). Larger values of α create more category nodes in the category representation layer of μARTMAP.

The resonance of a category (also mentioned in Step 1) is determined by examining if the function, called vigilance ratio, and defined below

$$\rho(\mathbf{I}, \mathbf{w}_j^a) = \frac{|\mathbf{I} \wedge \mathbf{w}_j^a|}{M_a} \tag{2}$$

satisfies the following condition:

$$\rho(\mathbf{I}, \mathbf{w}_j^a) \ge \rho_a$$

If the above equation is satisfied we say that resonance is achieved. The parameter ρ_a appearing in the above inequality is called vigilance parameter and assumes values in the interval [0, 1]. For Fuzzy ARTMAP, this parameter is globally applied to all categories. In μARTMAP, however, each category has its own ρ_a parameter, which is initialized as the global vigilance level when the category is committed and will not be changed afterwards; the global vigilance level is never used in the vigilance test, and it will not affect the existing categories' ρ_a parameters even when it is raised according to (5). At the beginning of training this parameter is set equal to a baseline vigilance level, designated by $\overline{\rho}_a$, which assumes values in the interval [0, 1], and is set by the user. After training commences the vigilance level is allowed to change and become larger than the baseline vigilance level, as categories in μARTMAP are destroyed for being too entropic (see Step 4 of the algorithm). Increased values of the vigilance level produce more nodes in the category representation layer of μARTMAP. If a chosen category j in μARTMAP passes the resonance test then this category is allowed to encode the presented input/associated label pair in the following manner:

$$\mathbf{w}_j^a = \mathbf{w}_j^a \wedge \mathbf{I} \qquad W_{jk}^{ab} = W_{jk}^{ab} + 1$$

where $k = label(\mathbf{I})$. The update of the templates, illustrated by the above equation, has been called *fast-learning* in the ART literature. The update of the inter-ART weights vector $\mathbf{W}_j^{ab}$ is such that the component of this weight that leads us to the correct label (i.e., $label(\mathbf{I})$) is increased by 1, while the rest of the components remain unchanged. So, during training the component W_{jk}^{ab} of the vector $\mathbf{W}_j^{ab}$ is equal to the number of times that category j encoded a pattern and this pattern had a k as its corresponding label (belonging to class k.)

If the category j is chosen and it resonates but the entropy of this category is higher than the allowable entropic threshold of a category, then this category is reset and the algorithm goes back to examine the rest of the categories to identify a new winner that resonates. Note that in this case the vigilance level is not increased and this is one of the differences between μARTMAP and the Fuzzy ARTMAP algorithms. The entropy of a category is defined by the following equation.

$$h_j = \frac{|\mathbf{W}_j^{ab}|}{|\mathbf{W}^{ab}|} entropy(j) = -\frac{|\mathbf{W}_j^{ab}|}{|\mathbf{W}^{ab}|} \sum_{k=1}^{N_b} \frac{\mathbf{W}_{jk}^{ab}}{|\mathbf{W}_j^{ab}|} \log_2 \left(\frac{W_{jk}^{ab}}{|\mathbf{W}_j^{ab}|} \right) \tag{3}$$

where

$$|\mathbf{W}_j^{ab}| = \sum_{k=1}^{N_b} W_{jk}^{ab} \qquad |\mathbf{W}^{ab}| = \sum_{j=1}^{N_a} \sum_{k=1}^{N_b} W_{jk}^{ab}$$

In the above equations $entropy(j)$ measures the entropy of node j, and h_j is the value of this entropy weighted by the relative frequency with which this node has encoded input/associated label pairs before. The level of this weighted entropy of the node that μARTMAP allows is a μARTMAP parameter value, denoted as $h_{\max}$, and it is value that has to be defined by the user.

In Safe μARTMAP, the winner category must also pass a distance test if it is already committed:

$$\frac{|\mathbf{w}_j^a| - |\mathbf{I} \wedge \mathbf{w}_j^a|}{M_a} \le \delta \tag{4}$$

where δ is also specified by the user, taking value from $(0, 1 - \overline{\rho}_a]$. This test requires that the size change of the winner category should not be too large due to a single pattern. If the winner category fails this test, no other categories will be picked to learn this pattern at this point. Instead, this pattern remains "unlearned". After all patterns are presented (which is called a *pass*), the unlearned patterns are presented again in the next *pass*. This time the previous winner categories may learn these patterns. If no pattern is learned in a whole pass, an unlearned pattern will be picked and a new category will be committed to learn this pattern; then all the other unlearned patterns are presented in the next pass. The above is repeated until all patterns are learned. In this way, the learning phase of a single epoch may consist of many passes. This is the only difference between Safe μARTMAP and the original μARTMAP.

After the 1st epoch of training is completed we examine the total entropy of the categories created. To do so we feed all the training input patterns to the trained μARTMAP architecture and we keep the count of how many times category j has been chosen by a pattern in the training set whose corresponding label is label k. In this cycle (referred to as off-line evaluation phase) a pattern chooses the category that receives the highest bottom-up input. At the end of this processing cycle we would have calculated a matrix $\mathbf{V}^{ab} = [V_{jk}^{ab}]$. The total entropy of the categories in μARTMAP is now defined as:

$$H = \sum_{j=1}^{N_a} h_j^{off} = \sum_{j=1}^{N_a} -\frac{|\mathbf{V}_j^{ab}|}{|\mathbf{V}^{ab}|} \sum_{k=1}^{N_b} \frac{V_{jk}^{ab}}{|\mathbf{V}_j^{ab}|} \log_2 \left(\frac{V_{jk}^{ab}}{|\mathbf{V}_j^{ab}|} \right)$$

If the total entropy H is larger than a pre-specified (by the user) threshold, designated as $H_{\max}$, then the total entropy is considered to be too high and the category j with the largest h_j^{off} is chosen, and destroyed. Then, the patterns that chose this category as their representative category in the learning phase of one of the previous epochs are presented again to μARTMAP. The global vigilance parameter level is increased to a value slightly higher than the vigilance ratio of the destroyed category; that is it is increased as:

$$\rho_a = \min \left(1, \qquad \frac{|\mathbf{I} \wedge \mathbf{w}_j^a|}{M_a} + \Delta\rho \right) \tag{5}$$

This way we are avoiding the creation of future categories in μARTMAP that are as entropic as the one that we have recently destroyed. As mentioned before, this new vigilance level will affect only the categories created in the next epoch (so that the destroyed category cannot be created again); the existing categories will keep their ρ_a values. The parameter $\Delta\rho$ is chosen to be a small positive constant. This process of offline evaluation of the total entropy, destruction of the most entropic category, and re-representation of the patterns that accessed this category in the epoch prior to the offline evaluation continues until we end up with a collection of categories whose total entropy does not exceed the designated allowable total entropic threshold of $H_{\max}$. At that time, we consider μARTMAP's training complete.

In all of the above equations there is a specific operand involved, called *fuzzy min operand*, and designated by the symbol $\wedge$. The fuzzy min operation applied on two vectors x and y, designated by $x \wedge y$, is a vector whose components are equal to the minimum of the corresponding components of x and y. Another specific operand involved in these equations is designated by the symbol $|\cdot|$. In particular, $|x|$ is the size of the vector x and it is defined to be the sum of its components.

As we have already mentioned an input pattern $\mathbf{I}$ presented at the F_1^a layer of μARTMAP has the following form:

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

where

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

The assumption here is that the input vector $\mathbf{a}$ is such that each one of its components lies in the interval $[0, 1]$. Any input pattern can be, through appropriate normalization, be represented by the input vector $\mathbf{a}$, where M_a stands for the dimensionality of the input pattern. The above operation that creates $\mathbf{I}$ from $\mathbf{a}$ is called complementary coding and it is required for the successful operation of μARTMAP.

The dimensionality of the input patterns in μARTMAP (i.e., M_a) is determined by the problem at hand. The same is true for the number of nodes (i.e., N_b) in the output layer of μARTMAP. The number of nodes N_a created in the category representation layer of μARTMAP is a parameter that is determined by the problem at hand and the values of the other μARTMAP parameters. μARTMAP has 5 parameter values that need to be specified before its training phase can be implemented. We have already talked about three of them (choice parameter α, and baseline vigilance parameter level $\bar{\rho}_a$, and the parameter $\Delta\rho$). These parameters are common in many ART architectures and their effect on the ART architectures has been studied. μARTMAP is introducing two more parameter values, i.e., $h_{\max}$ and $H_{\max}$, whose effect has been less studied, and consequently the setting of these parameters by the user becomes a more difficult task. In this paper we are explaining through some theoretical arguments how to choose $h_{\max}$ and $H_{\max}$, and we are validating this theory by performing a number of experimental results.

In the performance phase of μARTMAP, a test input is presented to the input layer of μARTMAP and the node in the category representation layer that receives the maximum bottom-up input is chosen (say node j). Then the predicted label for this test input is chosen to be the label that most often node j has been mapped to in μARTMAP's training process. That is the predicted label of this input is chosen to be the label *k* that maximizes W_{jk}^{ab}.

3. μARTMAP Parameters

3.1 Parameter α and e

The choice parameter α affects the competition of the nodes, according to (1). It is desired that:

1) if a point is inside two boxes, it should choose the smaller one;
2) if a point is inside one box, no matter how large it is, and outside another box with sufficient distance, it should choose the former one.

As a reminder the μARTMAP and Fuzzy ARTMAP represent the input data in terms of boxes (hyper-rectangles) that cover within their boundaries all the input patterns that chose this box as their representative box. Condition 1) requires simply $\alpha > 0$. Condition 2) cannot be satisfied if $|\mathbf{w}_j^a|$ can be arbitrarily small (or the box can be arbitrarily large). For the databases with input patterns, whose components, are normalized to lie in [0, 1], no positive α value allows a box to cover the whole input space (which means $|\mathbf{w}_j^a| = |\mathbf{I} \wedge \mathbf{w}_j^a| = 0$) and satisfies condition 2) at the same time. The authors of μARTMAP adjusted the algorithm by normalizing the input elements to the interval $[e, 1-e]$ instead of [0, 1] (Gomez-Sanchez, personal communication), and require that:

1) $\alpha << \min|\mathbf{w}_j^a| = 2M_a e$, so that when a point is inside a box, the corresponding T_j is close to one even if the box covers the whole input space.
2) $e << 1$, or otherwise the vigilance test would always pass when the vigilance parameter ρ_a is small, since $|\mathbf{I} \wedge \mathbf{w}_j^a| / |\mathbf{I}| \geq 2M_a e / M_a = 2e$

In our experiments, the choice parameter α was set to 0.01 and 0.001 for all ART algorithms the minimum M_a was 2. Due to the above constraints ($1/400 << e << 1$), we set e to 0.05 in our experiments. for the μARTMAP. We also did some preliminary experiments and found that the μARTMAP is not sensitive to α or to e as long as the above constraints are satisfied.

3.2 Parameter $H_{\max}$

The parameter $H_{\max}$ controls the impurity of the whole network. It terminates the training process to prevent over-training. $H_{\max}$ has a direct effect on the final accuracy of the μARTMAP. Setting $H_{\max} = 0$ means that the ARTMAP must have 100% accuracy on the training set in the offline evaluation, which is usually impractical. In most cases, $H_{\max} = 0$ not only keeps the training algorithm running for a long time, but also over fits the network to the training set, resulting in many trivial nodes that increase the network size and negatively affect the generalization (accuracy on unseen data). On the other hand, setting $H_{\max}$ to a very high value will terminate the training process too soon and result in low generalization, as well.

Apparently, the proper $H_{\max}$ value is problem-dependent. Nevertheless, we can come up with some estimates of the total entropy H. First, let N_b denote the number of classes (namely the number of nodes in the output layer), and $\hat{A}$ represent the expected accuracy given by the user and assumed in the interval $(1/N_b, 1]$. If there is a known theoretical optimal accuracy in a problem, assume $\hat{A}$ is equal to this theoretically optimal accuracy. Of course, $\hat{A}$ is sometimes unknown. Nevertheless, estimating $\hat{A}$ (using for example information existing in the literature) is much easier than guessing $H_{\max}$. Our experiments show that when the accuracy of the network on the training set reaches $\hat{A}$, the network tends to have the best accuracy on unseen patterns, as long as the training parameters are set properly. It can be proved that the entropy H is bounded as follows:

$$H_L \le H \le H_U$$
$$H_L = -\log_2 \hat{A}$$
$$H_U = -A \log_2 \hat{A} - \left(1 - \hat{A}\right) \log_2 \frac{1 - \hat{A}}{N_b - 1}$$

H_L is the entropy when $1/\hat{A}$ is an integer and the proportions of the classes in all categories are either 0 or $\hat{A}$. H_U is the entropy when the proportion of the major class in each category is $\hat{A}$ and the other classes are evenly distributed for all categories.

However, neither H_L nor H_U is a good estimate for $H_{\max}$, since both of them can be quite different from the actual entropy. Two other estimates for the entropy H are given below:

$$H_{E1} = \frac{(1 - \hat{A}) N_b \log_2 N_b}{N_b - 1}$$
$$H_{E2} = -\frac{N_b\left(1 - \hat{A}\right) - \left(N_b - 1\right)p}{1 - p} \log_2 p - \log_2 \hat{A}$$

where p is the solution in [0, 1] to the equation $\left(1 - p\right)/\left(1 - p^{N_b}\right) = \hat{A}$. H_{E1} is the entropy when the accuracies of all the categories are either 1 (pure categories) or $1/N_b$ (completely impure categories); H_{E2} is the entropy when the accuracies of all the categories are $\hat{A}$, and the proportion of the minor classes within each category forms a geometric progress.

In our experiments, we have used all these estimates of the entropy to come up with legitimate values of $H_{\max}$ to run our μARTMAP experiments.

3.3 Parameter $h_{\max}$

The parameter $h_{\max}$ controls the impurity of each node (category). A node may be both very large and very pure (which means most of the patterns that select it have the same class label). μARTMAP allows a large node to be created by allowing ρ_a to be zero, and maintains the accuracy by controlling the impurity. This is the main reason why μARTMAP can achieve a good accuracy with very few nodes.

The parameter $h_{\max}$ affects the training process mostly in the first epoch. From the second epoch, only the patterns learned by the category that is removed in the previous epoch will be presented. If these patterns are learned in exactly the same order as last time, then the entropy test will always pass no matter whether they go to a new category, since when each one of these patterns is learned, the total number of learned patterns $|\mathbf{W}^{ab}|$ is no less than last time, and $|\mathbf{W}_j^{ab}|\, entropy(j)$ will never increase when a pattern is removed from category j, due to the convexity of the entropy function. If the learning order is changed, due to the increased vigilance level, the entropy test may fail.

Setting $h_{\max} = 0$ means all the nodes must be completely pure when created or expanded; they may become impure as more patterns are presented and more nodes are created. This has the similar drawback as in $H_{\max} = 0$. Setting $h_{\max} = \infty$ means that the entropy test always passes.

According to (3), it is difficult to estimate a good $h_{\max}$ value, since $|\mathbf{W}_j^{ab}|$, the number of patterns learned by category j, is not easy to predict. Moreover, h_j is much more sensitive to the order in which the patterns are presented than the total entropy in the offline evaluation phase. For example, suppose there is only one category in the network, and the first four patterns it learned have class labels 1, 1, 1, 2. Its h_j would be 0, 0, 0, 0.8113 after it learned these patterns. If we swap the second and the fourth pattern, then its h_j would be 0, 1, 0.9183, 0.8113 after it learned these patterns. If we had set $h_{\max} = 0.9$, then this category could learned all the four patterns in the first case (before the swapping), but it could not learn the pattern with class label 2 in the second case.

Nevertheless, we assume the proper value of $h_{\max}$ is proportional to the proper value of $H_{\max}$. We expect the optimal $h_{\max}/H_{\max}$ ratio to be problem-dependent. Furthermore, it is very difficult to estimate this ratio, and definitely a lot more difficult than it was estimating $H_{\max}$. In our experiments, we varied the ratio between $h_{\max}$ and $H_{\max}$ in order to search for the optimal one.

3.4 Parameters $\overline{\rho}_a$, $\Delta\rho$, and δ

The baseline vigilance threshold $\overline{\rho}_a$ can be initialized as any value in [0, 1]. Only in the first epoch ρ_a explicitly depend on $\overline{\rho}_a$. From the start of the second epoch, ρ_a is determined by the size of the most entropic node and $\Delta\rho$, according to equation (5). However, it can be easily shown that $\rho_a \geq \overline{\rho}_a$ in all epochs, and thus $\overline{\rho}_a$ is still important for μARTMAP. For $\overline{\rho}_a = 0$ μARTMAP allows an arbitrarily large box to be created in the first epoch. For $\overline{\rho}_a = 1$ μARTMAP allows only zero-sized boxes. In our experiments, we varied $\overline{\rho}_a$ within the set of values {0, 0.2, 0.4, 0.6, 0.8}.

$\Delta\rho$ is introduced only to make sure the most entropic category cannot be created again after it is removed. Apparently, $\Delta\rho$ should not be set too high to avoid increasing ρ_a too quickly. If $\Delta\rho$ is too small, however, a category may be created with entropy close to that of the removed category, which means the total entropy may drop very slowly and it may take many epochs to finish μARTMAP's training. $\Delta\rho$ is also difficult to estimate since the size of the most entropic category in a certain epoch is obviously dependent on the distribution of the patterns. Nevertheless, our experiments showed that the performance of μARTMAP is not sensitive to $\Delta\rho$ as long as it is in a reasonable range. In our experiments, we fixed $\Delta\rho$ to the value of 0.02.

The parameter δ controls the *size change per pattern* of each category, according to (4). This parameter alleviates the overlapping problem in μARTMAP and reduces the effect of μARTMAP's dependence on the order of pattern presentation in the training set. Small δ means that the size change must be small. Usually it will cause longer training times because in each epoch, more patterns will be placed into the unlearned set for many passes, until they are finally learned. If $\delta = 0$, then no category can increase its size, which is equivalent to set $\overline{\rho}_a = 1$. If $\delta \geq 1 - \overline{\rho}_a$, then (4) is always satisfied, and Safe μARTMAP reduces to μARTMAP. The optimal δ value is also dependent on the distribution of patterns. Although δ makes the algorithm less sensitive to the order of pattern presentation in the training set, the optimal value of δ depends on the distribution of the data points more than the other parameters does.

4 Experiments

We have performed a number of experiments with μARTMAP. The purpose of these experiments was two-fold: First, to compare μARTMAP's performance with the performance of other ART classifiers in the literature, including ART architectures that claimed that they have also addressed the category proliferation problem in Fuzzy ARTMAP. Secondly, we have made an effort to identify "optimal" settings of the network parameters in μARTMAP. In the sequel, we are reporting results from both of these sets of experiments.

4.1 Databases

We experimented with both artificial and real databases. The specifics of these databases are given below.

1. Gaussian Databases (G#c-##)

   These are artificial databases, where we created 2-dimensional data, Gaussianly distributed, belonging to 2-class, 4-class, and 6-class problems. In each one of these databases we varied the amount of overlap of data belonging to different classes. In particular, we considered 5%, 15%, 25%, and 40% overlap. Note that 5% overlap means the optimal Bayesian Classifier would have 5% misclassification rate on the Gaussianly distributed data. There are a total of 3×4=12 Gaussian databases. We name the databases as "G#c-##" where the first number is the number of classes and the second number is the class overlap. For example, G2c-05 means the Gaussian database is a 2 class and 5% overlap database.
2. Modified Iris Database (MOD-IRIS)

In this database we started from the IRIS dataset (see Hettich et al, 1998) of the 150 3-class problem. We eliminated the data corresponding to the class that is linearly separable from the others. Thus we ended up with 100 data-points. From the 4 input attributes of this IRIS dataset we focused on only 2 attributes (attribute 3 and 4) because they seem to have enough discriminatory power to separate the 2-class data. Finally, in order to create a reasonable size dataset from these 100 points (so we can reliably perform cross-validation to identify the optimal μARTMAP parameters) we created noisy data around each one of these 100 datapoints (the noise was Gaussian of zero mean and small variance) to end up with approximately 10,000 points. We named this database Modified Iris.

3. Modified Abalone Database (ABALONE)

   This database is originally used for prediction of the age of an abalone (see Hettich et al, 1998). It contains 4177 instances, each with 7 numerical attributes, 1 categorical attribute, and 1 numerical target output (age). We discarded the categorical attribute in our experiments, and grouped the target output values into 3 classes: 8 and lower (class 1), 9-10 (class 2), 11 and greater (class 3). This grouping of output values has been reported in the literature before.

4. Page Blocks Database (PAGE)

   This database represents the problem of classifying the blocks of the page layout in a document (see Hettich et al, 1998). One of the noteworthy points about this database is that, its major class has a high probability of occurring (above 80%).

The data in each one of the above databases was split into a training set, a validation set, and a test set. The percentage of classes in each one of these subsets resembled the percentage of classes in the original dataset. The summarized specifics of each one of these databases are depicted in Table 1.

Database Name	# Training Instances	# Validation Instances	# Test Instances	# Numerical Attributes	# Classes (N_b)	% Major Class (A_0)	Expected Accuracy ($\hat{A}$)
G2c-05	500	5000	5000	2	2	1/2	0.95
G2c-15	500	5000	5000	2	2	1/2	0.85
G2c-25	500	5000	5000	2	2	1/2	0.75
G2c-40	500	5000	5000	2	2	1/2	0.6
G4c-05	500	5000	5000	2	4	1/4	0.95
G4c-15	500	5000	5000	2	4	1/4	0.85
G4c-25	500	5000	5000	2	4	1/4	0.75
G4c-40	500	5000	5000	2	4	1/4	0.6
G6c-05	504	5004	5004	2	6	1/6	0.95
G6c-15	504	5004	5004	2	6	1/6	0.85
G6c-25	504	5004	5004	2	6	1/6	0.75
G6c-40	504	5004	5004	2	6	1/6	0.6
MOD-IRIS	500	4800	4800	2	2	1/2	0.95
ABALONE	501	1838	1838	7	3	1/3	0.6
PAGE	500	2486	2487	10	5	0.832	0.95

Table 1: Databases used in the μARTMAP experiments

4.2 Parameter Settings:

For each database, we simulated Safe μARTMAP with all the following combinations of the five Safe μARTMAP parameters $H_{\max}, h_{\max}, \bar{\rho}_a, \alpha$ and δ.

$$H_{\max} = \{H_1,\ H_2,\ H_3,\ H_4,\ H_5\}$$

$$H_1 = \frac{1}{2}(H_L + H_2)$$

$$H_2 = \min\{H_{E1}, H_{E2}\}$$

$$H_3 = \frac{1}{2}(H_{E1} + H_{E2})$$

$$H_4 = \max\{H_{E1}, H_{E2}\}$$

$$H_5 = \begin{cases} H_U, & H_U > H_4 \\ 2H_U - H_3, & H_U = H_4 \end{cases}$$

$$h_{\max} = \left\{0,\ \frac{1}{4}H_{\max},\ \frac{1}{2}H_{\max},\ H_{\max},\ 2H_{\max}, \infty\right\}$$

$$\bar{\rho}_a = \left\{0,\ \frac{1}{5},\ \frac{2}{5},\ \frac{3}{5},\ \frac{4}{5}\right\}$$

$$\Delta\rho = 0.02$$

$$\alpha = \{0.001,\ 0.01\}$$

$$\delta = \left\{\frac{1}{25}(1-\bar{\rho}_a),\ \frac{1}{5}(1-\bar{\rho}_a),\ (1-\bar{\rho}_a)\right\}$$

$$e = 0.05$$

$$MaxNumberOfEpochs = 100$$

We experimented with all the combinations of the above parameters, which amounted to 5×6×5×2×3=900 combinations.

4.3 Experiment Procedure – Experimental Results

As we have emphasized above, our experiments were divided into two parts. In the first part, we compared Safe μARTMAP with other ARTMAP classifiers - Fuzzy ARTMAP, Ellipsoidal ARTMAP, Gaussian ARTMAP, distributed Gaussian ARTMAP, and their semi-supervised versions. For each database, we evaluated all the possible parameter combinations of Safe μARTMAP for a 100 different orders of the pattern list presentation. (the performance of μARTMAP depends on the order according to which patterns are presented in the training set). The 100 orders were fixed in all experiments and are exactly the same as those used to test the other ARTMAP algorithms. Therefore, for each database, we trained 900×100=90000 μARTMAP networks; we picked the network maximizing the following score:

$$score = \frac{A - A_0}{\hat{A} - A_0} 0.9^{(N_a/5N_b)^2}$$

where A is the accuracy on the validation set, N_a is the number of categories formed in the training phase of μARTMAP, and A_0, $\hat{A}$, and N_b are given in Table 1 . The accuracy is normalized to approximately [0, 1] so that we can compare the scores corresponding to different databases without bias. Apparently, the above score is monotonically increasing with A and monotonically increasing with N_a ; when N_a is small, $\partial score / \partial N_a \approx 0$.

The results are depicted in Table 2. In Table 2 we are showing the performance (in terms of accuracy on the test set and size) of the ART network that maximized the score value.

	Safe μAM		FAM		ssFAM		EAM		ssEAM		GAM		ssGAM		dGAM		ssdGAM	
G2c-05	95.22	2	90.80	14	94.90	2	91.72	26	94.94	2	94.06	4	94.48	4	95.22	4	95.22	2
G2c-15	85.00	2	77.68	47	84.80	3	77.88	79	85.20	2	84.86	6	85.04	2	84.76	8	85.02	2
G2c-25	74.98	2	64.36	75	74.60	2	65.06	123	74.50	2	74.88	6	75.10	2	74.90	7	75.10	2
G2c-40	61.40	3	53.84	110	61.34	3	53.58	177	60.98	2	59.64	12	61.30	3	60.30	9	61.32	3
G4c-05	95.04	4	92.84	21	94.10	7	92.96	24	94.14	4	94.84	10	94.80	4	94.84	10	94.80	4
G4c-15	83.28	4	77.52	55	81.40	11	78.12	76	83.20	4	84.00	18	84.24	9	84.00	18	84.20	9
G4c-25	74.50	4	67.06	101	70.80	9	66.58	110	72.72	4	73.74	49	72.32	21	74.60	46	74.96	35
G4c-40	59.76	5	48.52	127	58.48	14	49.58	161	55.62	13	58.08	36	59.10	14	58.92	36	59.40	14
G6c-05	93.57	9	91.85	26	91.42	11	92.30	23	93.80	7	94.49	12	94.40	8	94.68	13	94.84	6
G6c-15	80.92	6	76.23	58	81.11	7	76.09	85	81.80	6	84.67	19	84.35	13	85.03	19	83.87	11
G6c-25	70.74	13	66.66	87	69.62	15	63.74	124	71.10	7	73.24	30	72.86	20	73.65	32	73.22	20
G6c-40	58.03	11	51.40	196	56.35	17	50.69	193	54.21	17	58.51	70	55.65	13	59.03	70	55.50	13
MOD-IRIS	94.92	2	91.93	23	93.41	8	93.37	28	94.54	2	94.50	4	94.54	2	94.52	4	94.54	2

ABALONE	57.18	4	46.40	29	59.52	6	46.24	86	56.80	7	45.87	12	55.10	3	46.13	12	55.10	3
PAGE	88.82	6	83.27	10	90.63	3	76.71	34	89.54	3	85.52	9	89.34	5	85.52	9	89.34	5

μAM: Safe μARTMAP; FAM: Fuzzy ARTMAP; EAM: Ellipsoidal ARTMAP; GAM: Gaussian ARTMAP; dGAM: Distributed Gaussian ARTMAP; ss* : semi-supervised version

Table 2: Best Performance of All ART Algorithms

The above table shows that μARTMAP tends to yield a small network with high accuracy. As it can also be seen from the table, safe μARTMAP outperforms in terms of size Fuzzy ARTMAP, Ellipsoidal ARTMAP and Gaussian ARTMAP, and compares very favorably with ssFAM, ssEAM, and ssGAM and ssdGAM. Actually, the algorithms that produce as good results as safe μARTMAP are ssEAM and ssdGAM.

In the second part of our experiments, we elaborated on the search of the optimal parameter settings for Safe μARTMAP. For the Gaussian databases (for which we know the exact value of $\hat{A}$), we examined the parameters of the best networks we previously selected. For each parameter combination and each Gaussian database, we set the score of the parameter combination as the maximum score of the 100 networks trained with that parameter combination (these 100 networks correspond to the 100 different orders of the training patterns during Safe μARTMAP's training). Then, for every parameter combination we have 12 of these maximum scores (one maximum score for each of the 12 Gaussian datasets). We sum up these 12 maximum score numbers for every parameter combination, and then we rank these sums from highest to lowest. The 5 highest of these sums of maximum scores point us to the 5 default Safe μARTMAP that we chose as a good set of parameters to experiment with, for any database of interest. To verify our claim, that the thus chosen 5 sets of default parameter values are good sets of parameters to experiment with, we are showing (in Table 3) Safe μARTMAP's performance (number of categories and accuracy on the test set) for the best Safe μARTMAP parameter values (set of columns designated as Best in the table) and for the 5 default parameter values that were identified from our experimentation of the Gaussian datasets, and explained above. Each performance cell in Table 3 is made up of three numbers: the accuracy on the test set in percentage, the number of categories, and the number of epochs spent on training. An obvious observation, as we compare the results of Safe μARTMAP's performance for the best parameter setting (which is database dependent and as such very time consuming to produce), and Safe μARTMAP's performance for the 5 default parameter settings, is that the default parameters produce good results. It is also important to know that the identification of good, default parameter values for Safe μARTMAP is saving us significant computations when Safe μARTMAP is used with a new database. Furthermore, the identification of good, default parameter values is essential in cases where the number of data-points in our dataset is not large enough to allow us the luxury of splitting the data into training and validation sets and performing cross-validation using the validation set.

Rank	Best			1			2			3			4			5		
$H_{\max}$	-			H_4			H_4			H_4			H_4			H_4		
$h_{\max}/H_{\max}$	-			∞			∞			1			∞			∞		
$\bar{\rho}_a$	-			0.4			0			0.2			0			0.2		
α	-			0.001			0.01			0.001			0.001			0.001		
$\delta/(1-\bar{\rho}_a)$	-			0.2			0.2			0.2			1			1		
G2c-05	95.22	2	1	95.16	2	14	95.14	2	4	95.20	2	1	95.20	3	10	95.20	3	10
G2c-15	85.00	2	1	85.06	2	4	84.98	3	15	85.06	2	1	85.24	2	27	85.24	2	27
G2c-25	74.98	2	1	74.96	2	16	74.96	3	18	74.18	2	1	75.02	3	8	75.02	3	8
G2c-40	61.40	3	1	61.54	4	8	61.34	4	18	61.44	3	10	61.32	4	32	61.32	4	32
G4c-05	95.04	4	22	94.82	4	25	94.36	6	50	94.64	4	1	94.46	6	48	94.46	6	48

G4c-15	83.28	4	20	81.74	6	44	84.18	7	65	83.58	9	82	83.64	9	61	83.64	9	61
G4c-25	74.50	4	44	74.78	5	37	75.06	6	52	75.06	4	48	75.02	6	49	75.02	6	49
G4c-40	59.76	5	39	59.26	4	52	59.76	5	39	58.84	5	41	59.72	7	37	59.72	7	37
G6c-05	93.57	9	9	93.09	10	85	91.87	9	74	93.23	10	58	93.53	13	93	93.53	13	93
G6c-15	80.92	6	1	81.18	12	100	81.87	13	100	81.16	14	76	82.27	12	100	82.27	12	100
G6c-25	70.74	13	88	71.18	13	83	69.54	14	85	69.76	11	100	69.16	13	90	69.16	13	90
G6c-40	58.03	11	100	56.77	16	100	56.45	13	81	56.41	13	100	56.30	14	77	56.30	14	77
MOD-IRIS	94.92	2	2	94.92	4	10	95.15	4	19	94.92	4	16	94.63	3	10	94.63	3	12
ABALONE	57.18	4	4	55.06	2	2	54.08	2	4	54.52	3	2	53.59	2	6	53.59	2	6
PAGE	88.82	6	17	88.34	5	10	92.32	5	24	89.14	8	35	89.75	4	11	89.75	4	11

Table 3: Best Parameter Combinations

Although the networks were ranked by cross-validation, the accuracy on the validation set is not shown, because it is always close to the accuracy on the test set.

According to the above table, we obtained the following optimal settings, assuming the maximum number of epochs is large enough:

$$H_{\max} = H_4, \;\; h_{\max} = \infty, \;\; \overline{\rho}_a = 0, \;\; \alpha = 0.001$$

We do not claim an optimal δ value because it depends on the size of the training set (and the relationship is not clear yet). For $H_{\max}$, we are very confident since all the best 5 parameter combinations have this value. In fact, all the best 65 networks have $H_{\max} = H_4$. This result is not surprising, since H_4 is a good estimate of the entropy without over-training. $\overline{\rho}_a = 0$ means we should allow a category to be very large in the first epoch, which is one of the benefits of µARTMAP. $\alpha = 0.001$ agrees with the results for the other ARTMAP architectures.

Although the optimal value of $h_{\max}$ seems unexpected, it can be explained as follows. This value allows a category to be very impure and tends to result in many more epochs of training because many impure categories must be removed in the future. In the first epoch, large categories will be created due to the small $\overline{\rho}_a$ value. In only a few epochs, the size of the categories will be controlled by ρ_a only. The number of categories will be very small in the beginning and it will grow slowly afterwards, until the total entropy is no more than $H_{\max}$. Therefore, the minimum number of categories may be achieved. Of course, sufficient epochs of training must be allowed, or otherwise the training process would be terminated prematurely and the network performance would be even worse than when $h_{\max} = 0$. In contrast, setting $h_{\max} = 0$ will cause a large number of categories to be created in the first epoch, including many trivial categories. In this case, the training process may finish in only one epoch, resulting in a network that may still be over-trained, exhibiting poor generalization.

5 Summary

Safe µARTMAP is one of the advanced ARTMAP architectures, which can produce small size classifiers with high accuracy. The main issue of using µARTMAP is the correct selection of its many parameters. In this paper, we studied the effect of the parameters, both theoretically and experimentally. Furthermore, we have identified a procedure that came up with a way of choosing good default µARTMAP parameter values, independently of the database used, despite the obvious fact that the best µARTMAP parameter values are data-base dependent. This is a significant simplification for anyone experimenting with µARTMAP on new datasets. Furthermore, it is also very beneficial in cases when the dataset is small and we do not have the option of splitting the dataset in training and validation sets. Also, we compared the performance of µARTMAP with a number of ART classifiers, including a number of them that have been reported in the literature and claim that they also address the category proliferation problem in Fuzzy ARTMAP. The result from this experimentation is that µARTMAP outperforms Fuzzy ARTMAP (FAM), Ellipsoidal ARTMAP (EAM), and Gaussian ARTMAP (GAM), and it exhibits comparable performance with semi-supervised EAM and distributed GAM. Finally, it is worth pointing out that our performance comparison of various ART algorithms and the identification of good, default parameter values for μARTMAP relied on a performance measure (score) that takes into consideration both the accuracy of the network on a cross-validation set and the size of the network that training creates. Despite its obvious benefits this is an approach that has not been quantified in the ART literature before.

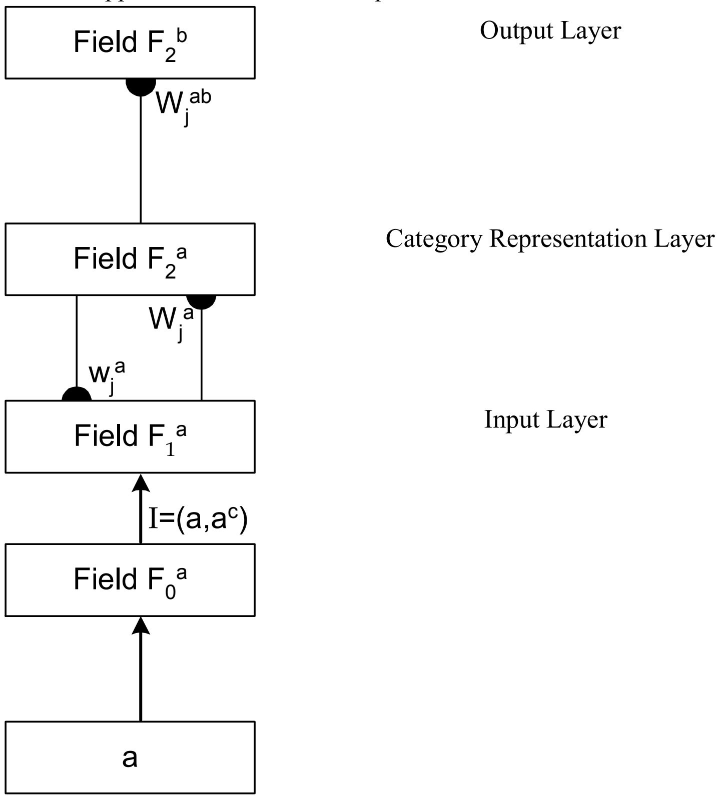

Figure 1: The block diagram of a μARTMAP Architecture

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

References

Anagnostopoulos, G., and Georgiopoulos, M., "Ellipsoid ART and ARTMAP for incremental clustering and classification," *IEEE-INNS International Joint Conference on Neural Networks 2001 (IJCNN 2001)*, Washington, DC, July 14-19, 2001, pp. 1221-1226.

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.

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, 2003, Volume 4, pp. 2782-2787.

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, 1992, pp. 698-713.

Gomez-Sanchez, E., Dimitriadis, Y. A., Cano-Izquierdo, J. M., and Lopez-Coronado, J., "μARTMAP: use of mutual information for category reduction in Fuzzy ARTMAP", *IEEE Transactions on Neural Networks*, Vol. 13, No. 1, Jan. 2002, pp. 58-69.

Gomez-Sanchez, E., Dimitriadis, Y. A., Cano-Izquierdo, J. M., Lopez-Coronado, J., "Safe- μARTMAP: a new solution for reducing category proliferation in Fuzzy ARTMAP," *IEEE Proceedings International Joint Conference on Neural Networks (IJCNN) 2001*, Vol. 2, July 15-19, 2001, pp. 1197-1202.

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

Hettich, S. & Blake, C.L. & Merz, C.J. *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

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

Williamson, J. R., "A constructive, incremental-learning network for mixture modeling and classification," Neural Computation, Vol. 9, 1997, pp. 1517-1543.

Verzi, S., Georgiopoulos, M., Heileman, G., and Healy, M., "Rademacher penalization applied to Fuzzy ARTMAP and Boosted ARTMAP, " *IEEE-INNS International Joint Conference on Neural Networks 2001 (IJCNN 2001)*, Washington, DC, July 14-19, 2001, pp. 1191-1196.