

An Ordering Algorithm for Pattern Presentation in Fuzzy ARTMAP that Tends to Improve Generalization Performance

I. Dagher*, M. Georgiopoulos*, G. L. Heileman**, G. Bebis***

*Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816

Phone: (407) 823-5338 Fax: (407) 823-5835
ijd, mng@ece.engr.ucf.edu

**Department of Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM 87131

Phone: (505) 277-4011 Fax: (505) 277-1439
heileman@eece.unm.edu

***Department of Computer Science
University of Nevada-Reno
Reno, NV 89557

Phone: (702) 784-6463 Fax: (702) 784-1877
bebis@cs.unr.edu

Abstract

In this paper we introduce a procedure, based on the Max-Min clustering method, that identifies a fixed order of training pattern presentation for Fuzzy ARTMAP. This procedure is referred to as the Ordering Algorithm, and the combination of this procedure with Fuzzy ARTMAP is referred to as Ordered Fuzzy ARTMAP. Experimental results demonstrate that Ordered Fuzzy ARTMAP exhibits a generalization performance that is better than the average generalization performance of Fuzzy ARTMAP, and in certain cases as good as, or better than the best Fuzzy ARTMAP generalization performance. We also calculate the number of operations required by the Ordering Algorithm and compare it to the number of operations required by the training phase of Fuzzy ARTMAP. We show that, under mild assumptions, the number of operations required by the Ordering Algorithm is a fraction of the number of operations required by Fuzzy ARTMAP.

1 Introduction

Pattern classification is a key element in many engineering applications. For example, sonar, radar, seismic, and diagnostic applications all require the ability to accurately classify data. In addition, control, tracking and prediction systems will often use classifiers to determine input-output relationships. Simpson has identified a number of desirable properties that a pattern classifier should possess [1]. These properties can be summarized as follows: A successful pattern classifier should be able to (i) learn the required task quickly, (ii) learn new data without having to retrain

with old data (on-line adaptation), (iii) solve non-linearly separable problems, (iv) provide the capability for soft and hard decisions regarding the degree of membership of the data within each class, (v) offer explanations of how the data are classified, and why the data are classified as such, (vi) exhibit performance that is independent of parameter tuning, (vii) function without knowledge of the distributions of the data in each class, and (viii) for overlapping pattern classes, create regions in the space of the input parameters that exhibit the least possible overlap.

A neural network classifier that satisfies most of the aforementioned properties is Fuzzy ARTMAP [2]. Fuzzy ARTMAP is capable of establishing arbitrary mappings between an analog input space of arbitrary dimensionality and an analog output space of arbitrary dimensionality. Fuzzy ARTMAP is a member of the class of neural network architectures referred to as *ART-architectures* developed by Carpenter, Grossberg, and colleagues. The ART-architectures are based on the *ART theory* introduced by Grossberg [3].

Fuzzy ARTMAP can operate in *off-line* or *on-line* modes. In the on-line mode, the network must process the data as it becomes available, without storing or reusing it. In the off-line mode, the data can be stored and repeatedly presented to the network. In this paper, we consider the off-line operation of Fuzzy ARTMAP in *classification problems* (e.g., recognizing handwritten digits). In particular, we consider one of the major limitations of Fuzzy ARTMAP, its dependence on tuning parameters (which is a violation of property (vi) above). It has been documented in the literature that the performance of Fuzzy ARTMAP depends on the values of two parameters called the choice and vigilance parameters, and also on the order of pattern presentation for the off-line mode of training. To circumvent the first problem, most Fuzzy ARTMAP simulations that have appeared in the literature assume zero values for the choice and vigilance parameters. One of the main reasons for the popularity of this choice is that it tends to minimize the size of the resulting network architecture. This is quite desirable, especially when performance comparisons are made between Fuzzy ARTMAP and other neural network architectures that offer more compact representations of the data, such as multi-layer perceptrons [4]. The problem of pattern ordering is not as easy to solve. One way around it is to consider different orders of presentations of the training data, in order to find the one that maximizes the performance of the network. The drawbacks of this approach include the considerable experimentation that is required to find a random order of pattern presentation that achieves a good network performance, and the fact

that this is essentially a guessing exercise. In this paper, we preprocess the training data by applying a systematic procedure (based on the Max-Min clustering algorithm [5]), which identifies a fixed order of pattern presentation. We refer to this procedure as the *Ordering Algorithm*. When the training input patterns are presented to Fuzzy ARTMAP according to this fixed order we end up with a trained Fuzzy ARTMAP whose generalization performance is better than the average generalization performance of Fuzzy ARTMAP, and in certain cases as good as, or better than the best network generalization performance. In the former case we consider the average of a fixed number of experiments corresponding to random orders of training pattern presentations, and in the latter case we consider the best of a fixed number of experiments corresponding to random orders of training pattern presentations. For simplicity, we refer to Fuzzy ARTMAP trained with the fixed order of input pattern presentations as *Ordered Fuzzy ARTMAP*. Ordered Fuzzy ARTMAP has the following desirable properties:

1. It achieves good generalization performance without requiring parameter tuning;
2. The sizes of the networks that Ordered Fuzzy ARTMAP creates are comparable to the sizes of the networks that Fuzzy ARTMAP creates when trained using a random order of pattern presentation;
3. Under mild conditions, the computational overhead imposed by the Ordering Algorithm is small compared to the computations required to perform the training phase of Fuzzy ARTMAP for a single random order of pattern presentation.

The organization of the paper is as follows. In Section 2, we briefly discuss the Fuzzy ARTMAP architecture, including the form of its inputs, training phase, performance phase, and functionality. In Section 3, we introduce the Ordering Algorithm. In Section 4, we illustrate with simple examples the effect of the Ordering Algorithm on the categories created by Fuzzy ARTMAP, and we explain the motivation for choosing this Ordering Algorithm. In Section 5, we experimentally demonstrate the superiority of Ordered Fuzzy ARTMAP's generalization performance compared to the average Fuzzy ARTMAP generalization performance, and in certain cases compared to the best Fuzzy ARTMAP generalization performance. Also, in Section 5 we discuss the computational complexity of the Ordering Algorithm and compare it to the computational complexity of the training phase of Fuzzy ARTMAP. Finally, in Section 5 we

compare the generalization performance of the Ordered Fuzzy ARTMAP and other classification techniques that have appeared in the literature. Section 6 provides a summary of the paper and offers some concluding remarks.

2 The Fuzzy ARTMAP Neural Network

A detailed description of the Fuzzy ARTMAP neural network can be found in [2]; we present only the necessary details here. The Fuzzy ARTMAP neural network consists of two Fuzzy ART modules, designated ART_a and ART_b , as well as an inter-ART module as shown in Figure 1. Inputs are presented at the ART_a module, while their corresponding outputs are presented at the ART_b module. The inter-ART module includes a MAP field whose purpose is to determine whether the correct input-output mapping has been established. The input pattern, designated by \mathbf{I} has the form

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1, \dots, a_{M_a}, a_1^c, \dots, a_{M_a}^c) \quad \text{where} \quad a_i \in [0, 1] \quad \text{and} \quad a_i^c = 1 - a_i; \quad 1 \leq i \leq M_a \quad (1)$$

The output pattern, designated by \mathbf{O} , has the form

$$\mathbf{O} = (b_1, \dots, b_{M_b}) \quad \text{where} \quad b_k \in [0, 1]; \quad 1 \leq k \leq M_b \quad (2)$$

Fuzzy ARTMAP operates in two distinct phases: the *training phase* and the *performance phase*. As mentioned earlier, in this paper we are interested in the off-line operation of Fuzzy ARTMAP in classification tasks, where many inputs are mapped to a single, distinct output. The *off-line training phase* of Fuzzy ARTMAP works as follows: Given a list of training input/output pairs, such as $\{\mathbf{I}^1, \mathbf{O}^1\}, \dots, \{\mathbf{I}^r, \mathbf{O}^r\}, \dots, \{\mathbf{I}^{PT}, \mathbf{O}^{PT}\}$, we want to train Fuzzy ARTMAP to map every input pattern of the training list to its corresponding output pattern. In order to achieve the aforementioned goal, we present the training list repeatedly to the Fuzzy ARTMAP architecture. That is we present \mathbf{I}^1 to ART_a and \mathbf{O}^1 to ART_b , then \mathbf{I}^2 to ART_a and \mathbf{O}^2 to ART_b , \dots , and finally \mathbf{I}^{PT} to ART_a and \mathbf{O}^{PT} to ART_b . This corresponds to *one list presentation*. The training list is presented as many times as it is necessary for Fuzzy ARTMAP to correctly classify all the input patterns. The classification task is considered accomplished (i.e., learning is complete) when the weights do not change during a list presentation. The *performance phase* occurs when the trained Fuzzy ARTMAP network is used to classify a list of test input patterns.

Two of the most important network parameters of the Fuzzy ARTMAP architecture are the choice parameter and the vigilance parameter in the ART_a module. For all of the experiments in this paper the values of these parameters are chosen to be zero (actually the choice parameter is a very small positive constant), because our objective is to focus on algorithms that do not require tuning of parameters. As mentioned previously, this choice of the network parameters leads to Fuzzy ARTMAP network architectures of minimum size. The functionality of Fuzzy ARTMAP is better illustrated by referring to the geometrical interpretation of the weights in ART_a . As initially discussed in [6] and further elaborated in [7], every weight vector in ART_a defines a hyperrectangle (hyperbox) in the input pattern space that includes all patterns that chose this weight vector as their representative during the training process. In Figure 2, we show the hyperrectangle that the weight vector \mathbf{w}_j^a (i.e., the weight vector corresponding to node j in F_2^a) defines. Note that patterns $\mathbf{I}^1 = (\mathbf{a}^1, (\mathbf{a}^1)^c)$, $\mathbf{I}^2 = (\mathbf{a}^2, (\mathbf{a}^2)^c)$, $\mathbf{I}^3 = (\mathbf{a}^3, (\mathbf{a}^3)^c)$, $\mathbf{I}^4 = (\mathbf{a}^4, (\mathbf{a}^4)^c)$, and $\mathbf{I}^5 = (\mathbf{a}^5, (\mathbf{a}^5)^c)$ were coded by $\mathbf{w}_j^a = (\mathbf{u}_j^a, (\mathbf{v}_j^a)^c)$, where \mathbf{u}_j^a and \mathbf{v}_j^a correspond to the endpoints of the hyperrectangle that \mathbf{w}_j^a defines. In Figure 2, hyperrectangles are actually rectangles since the input patterns \mathbf{I} are 4-dimensional, and their components (the \mathbf{a} 's) are 2-dimensional. After the training of Fuzzy ARTMAP is completed, the weight vectors of committed nodes in ART_a represent clusters of input patterns (hyperboxes) that are mapped to the same output pattern.

3 The Ordering Algorithm

The purpose of the Ordering Algorithm of Ordered Fuzzy ARTMAP is to identify the order in which patterns should be presented during the training phase of Fuzzy ARTMAP. This task is accomplished by following a systematic procedure that consists of three stages. Before we discuss these stages let us first define the parameters n_{clust} , PT , and the set S_T that appear in the algorithm's description. In this paper, the parameter n_{clust} is taken to be either equal to the number of distinct classes or equal to one more than the number of distinct classes associated with the pattern classification task. The parameter PT stands for the number of input/output pairs in the training list. Finally, S_T is the set of all training input patterns prior to the application of the Ordering Algorithm S_T . In Stage 1, we choose the first pattern to be presented. This pattern corresponds to the first cluster center of the training

input patterns. In Stage 2, we choose the next $(n_{clust} - 1)$ patterns to be presented. These patterns correspond to the next $(n_{clust} - 1)$ cluster centers of the training input patterns, and are identified using the *Max-Min* clustering algorithm [5]. In Stage 3, we choose the remaining $(PT - n_{clust})$ patterns to be presented. These patterns are chosen according to the minimum Euclidean distance criterion from the n_{clust} cluster centers defined in Stages 1 and 2. Below, we describe in more detail each of these stages.

Stage 1: *The first pattern*

For each pattern $\mathbf{I} = (a_1, \dots, a_{M_a}, a_{M_a+1}, \dots, a_{2M_a})$ in the training set we compute

$$\sum_{i=1}^{M_a} |a_{M_a+i} - a_i| \quad (3)$$

The pattern from the training set that maximizes the above sum is the first pattern presented to Ordered Fuzzy ARTMAP, and the first cluster center used in Stage 2. The training pattern that maximizes the above sum is removed from the training set S_T . To understand how the first pattern is produced in the first stage of the Ordering Algorithm we present a simple example in Appendix 1. The following two stages of the ordering procedure involve calculation of Euclidean distances among patterns in the training set. In the calculation of these distances only the first M_a components of the input patterns are used (i.e., the \mathbf{a} portion of the \mathbf{I} vector). To avoid switching back and forth between the \mathbf{a} and \mathbf{I} notation, we refer to these distances as the distances among the \mathbf{I} 's.

Stage 2: *The next $(n_{clust} - 1)$ patterns*

This stage uses the Max-Min clustering algorithm to define $(n_{clust} - 1)$ appropriate cluster centers (patterns), which constitute the next $(n_{clust} - 1)$ input patterns to be presented during the training phase of Ordered Fuzzy ARTMAP. The steps followed to define these cluster centers are as follows. The index r , initialized and updated in the step-by-step description of Stage 2, corresponds to the number of clusters that have been identified, at various points, during the implementation of Stage 2.

1. Denote the first cluster center (input pattern) identified in Stage 1 by \mathbf{I}_O^1 , and initialize the index r to one.

2. Compute the Euclidean distance of every input pattern in the training set S_T to the k -th cluster center, and find the minimum one, d_{min}^k . That is,

$$d_{min}^k = \min_{\mathbf{I} \in S_T} \{dist(\mathbf{I}, \mathbf{I}_O^k)\} \quad (4)$$

Repeat the above step for all cluster centers k , such that $1 \leq k \leq r$.

3. Find the input pattern from the training set S_T that maximizes d_{min}^k , $1 \leq k \leq r$. Designate this input pattern by the generic name \mathbf{I} . The next cluster center, designated by \mathbf{I}_O^{r+1} is equal to \mathbf{I} , that is $\mathbf{I}_O^{r+1} = \mathbf{I}$. This cluster center constitutes the next input pattern to be presented during the training phase of Ordered Fuzzy ARTMAP. Increment r by one, and eliminate input pattern \mathbf{I} from the training set S_T .
4. If $r = n_{clust}$ this stage is completed; otherwise, go to Step 2.

At the end of Stages 1 and 2, we have identified n_{clust} cluster centers that correspond to the input patterns \mathbf{I}_O^r , $1 \leq r \leq n_{clust}$, of the training set. The next stage identifies the order according to which the remaining input patterns should be presented to the Ordered Fuzzy ARTMAP.

Stage 3: *The remaining $(PT - n_{clust})$ input patterns*

The steps followed in this stage are as follows:

1. Set index r to n_{clust} . The patterns in the training set S_T are all of the training input patterns except the ones identified as cluster centers in Stages 1 and 2.
2. Calculate the Euclidean distance of every pattern \mathbf{I} in the set S_T to the n_{clust} cluster centers.
3. Find the minimum of these distances. Assume that it corresponds to input pattern \mathbf{I} . This pattern is the next in sequence input pattern to be presented in the training phase of Fuzzy ARTMAP. Eliminate \mathbf{I} from the set S_T , set $\mathbf{I}_O^{r+1} = \mathbf{I}$, and increment r .
4. If $r = PT$ this stage is complete; otherwise, go to Step 2.

After the end of Stage 3, we have identified the ordered set of patterns $\mathbf{I}_O^1, \mathbf{I}_O^2, \dots, \mathbf{I}_O^{PT}$. This is the order according to which the patterns in the training set will be presented to the Ordered Fuzzy ARTMAP. The corresponding outputs of this ordered sequence of input patterns are the outputs from the training list that these input patterns need to be mapped to. For example, if $\mathbf{I}_O^1 = \mathbf{I}^2$, then \mathbf{I}_O^1 's corresponding output is \mathbf{O}^2 . It is worth mentioning that the ordering that the Ordering Algorithm produces is independent of any permutations of the input training patterns. Proof of this statement is provided in Appendix 2 (Theorem 1).

4 Examples – Motivation

In order to better understand the differences between a random order of training pattern presentation and the proposed fixed order of training pattern presentation, we present some illustrative examples. Once the examples are presented it will be easier to explain the motivation for our work. In Example 1 (see Figure 3) the data $\{0, 0.1, 0.4, 0.5\}$ belong to Class 1, while the data $\{0.2, 0.3, 1\}$ belong to Class 2. The Ordering Algorithm, with $n_{clust} = 2$, computed the following order for training pattern presentation $\{0, 1.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Note that Stage 1 in this example identified pattern 0 as the first pattern to be presented (the closest point to one of the corners of the input pattern space), Stage 2 identified pattern 1 as the second pattern to be presented, and Stage 3 identified the order for the rest of the patterns in the training set. The training patterns in Fuzzy ARTMAP were presented in the following order $\{1.0, 0.2, 0.3, 0, 0.1, 0.4, 0.5\}$. Observe that in Figure 3, the numbers above the black and gray circles indicate the order of their presentation in the training phases of Fuzzy ARTMAP and Ordered Fuzzy ARTMAP. After training is over in Fuzzy ARTMAP, rectangles R_1 (with endpoints 0.2 and 1), R_2 (with endpoints 0 and 0.5), and R_3 (with endpoints 0.2 and 0.3) have been created. After training is over in Ordered Fuzzy ARTMAP, rectangles R_1 (with endpoints 0 and 0.1), R_2 (containing the datum 1.0), R_3 (with endpoints 0.2 and 0.3), and R_4 (with endpoints 0.4 and 0.5) have been created. The decision regions for Fuzzy ARTMAP (shown in Figure 3), classify test data between 0.2 and 0.3 and between 0.5 and 1, as Class 2, and the rest of the test data as Class 1. On the other hand, the decision regions for the Ordered Fuzzy ARTMAP (shown in Figure 3) classify the test data between 0 and 0.15 and between 0.35 and 0.74 as Class 1, and the rest of the data as Class 2. In this example, Ordered Fuzzy ARTMAP has

reduced the effect of the non-representative rectangle R_1 . This reduction was done by shrinking the decision region $[0.5, 1]$ to the region $[0.74, 1]$. Rectangle R_1 of Fuzzy ARTMAP is non-representative of the data because it forces the network to classify all test data in the interval $[0.5, 1]$ as data belonging to Class 2, despite the fact that there is no Class 2 training data within the interval $[0.3, 1]$.

In Example 2 (see Figure 4), the data $\{0, 0.05, 0.1, 0.4, 0.45, 0.51\}$ belong to Class 1, while the data $\{0.2, 0.25, 0.3, 0.6, 0.65, 0.7\}$ belong to Class 2. The Ordering Algorithm, with $n_{clust} = 2$, identified the following order of training pattern presentation $\{0, 0.7, 0.05, 0.65, 0.1, 0.6, 0.51, 0.2, 0.25, 0.45, 0.3, 0.4\}$ for Ordered Fuzzy ARTMAP. Note that Stage 1 in this example identified pattern 0 as the first pattern to be presented (the closest point to one of the corners of the input pattern space), Stage 2 identified pattern 0.7 as the second pattern to be presented, and Stage 3 identified the order for the rest of the patterns in the training set. The training patterns in Fuzzy ARTMAP were presented in the following order $\{0, 0.51, 0.15, 0.1, 0.2, 0.25, 0.3, 0.4, 0.45, 0.6, 0.65, 0.7\}$. Observe that in Figure 4, the numbers above the black and gray circles indicate the order of their presentation during the training phases of Fuzzy ARTMAP and Ordered Fuzzy ARTMAP. After training is over in Fuzzy ARTMAP, rectangles R_1 (with endpoints 0 and 0.51), R_2 (with endpoints 0.2 and 0.7), and R_3 (with endpoints 0.4 and 0.51) have been created. After training is over in Ordered Fuzzy ARTMAP, rectangles R_1 (with endpoints 0 and 0.1), R_2 (with endpoints 0.6 and 0.7), R_3 (with endpoints 0.2 and 0.3), and R_4 (with endpoints 0.4 and 0.51) have been created. The decision regions for Fuzzy ARTMAP (shown in Figure 4) classify test data between 0 and 0.2 and between 0.4 and 0.51 as Class 1, and the rest of the data as Class 2. On the other hand, the decision regions for the Ordered Fuzzy ARTMAP (shown in Figure 4) classify the test data between 0 and 0.15 and the test data between 0.35 and 0.55 as Class 1, and the rest of the data as Class 2. In this example, Ordered Fuzzy ARTMAP has reduced the overlap among rectangles that lead to different outputs (see desired Property (viii) of a pattern classifier).

The major motivation for our work was the design of a Fuzzy ARTMAP algorithm that is independent of the tuning of parameters, and achieves good generalization by avoiding excessive experimentation. The dependence of Fuzzy ARTMAP on the choice parameter and the vigilance parameter is an inherent characteristic of the algorithm. Choosing these parameters equal to zero frees the experimenter from the tedious task of optimizing the network

performance with respect to these two parameters. With the choice parameter and the vigilance parameter chosen equal to zero, one ends up with a Fuzzy ARTMAP algorithm that exhibits a significant variation in generalization performance for different orders of training pattern presentations. Furthermore, it is not an easy task to guess which one of the exceedingly large number of orders of pattern presentations exhibits the best generalization. Our assumption was that orders of pattern presentation that create unnecessarily large rectangles (e.g., rectangle R_1 of Example 1 for Fuzzy ARTMAP), that force assumptions about the data where training data do not exist (e.g., region $[0.5, 1]$ for Fuzzy ARTMAP in Example 1), give credibility to possible outlier data (e.g., datum “1” in Example 1 for Fuzzy ARTMAP), and include in their regions data that belong to more than one class (e.g., Examples 1 and 2), would not lead to good generalization performance. So, the idea was to address this problem by spreading out enough initial clusters at locations where data exist (e.g., clusters 1 and 2 in Examples 1 and 2), and then present the rest of the training data in order of closest distance to these initial clusters. The examples presented above justified the validity of our approach.

5 Experimental Results – Comparisons

In the following sections, we describe the databases used to compare Fuzzy ARTMAP and Ordered Fuzzy ARTMAP, define performance measures, and compare the two algorithms by conducting appropriate experiments.

5.1 Databases

In order to demonstrate the superior performance of Ordered Fuzzy ARTMAP as compared to Fuzzy ARTMAP, we chose to conduct experiments on a number of databases extracted from the UCI repository database [8]. The databases chosen from the repository were: Iris [9], Wine [10], Sonar [11], Diabetes [12], Breast [13], Balance [14], Bupa [8], Cars [8], and Glass [8]. The Sonar, Diabetes, Breast, and Bupa are two-class classification problems, the Iris, Wine, Balance are three-class classification problems, the Cars is a four-class classification problem, and finally the Glass is a six-class classification problem. Each of these datasets was split randomly into a training set (2/3

of the data) and a test set (1/3 of the data). The percentage of data from each class in the training and test set reflected the percentage of data from each class in the entire dataset. The number of the data used in the training set for Iris, Wine, Sonar, Diabetes, Breast, Balance, Bupa, Cars and Glass were 102, 120, 139, 513, 467, 417, 231, 566, 145, respectively. The number of the data used in the test set for Iris, Wine, Sonar, Diabetes, Breast, Balance, Bupa, Cars and Glass were 48, 58, 69, 255, 232, 208, 114, 280, 69, respectively. The dimensionality of the input patterns for Iris, Wine, Sonar, Diabetes, Breast, Balance, Bupa, Cars and Glass were 4, 13, 60, 8, 9, 4, 6, 18, 9, respectively. More detailed descriptions of each one of these databases can be found in the references.

5.2 Measures of Performance

One of the performance measures used to compare Ordered Fuzzy ARTMAP and Fuzzy ARTMAP is *generalization*. The generalization performance of a network is defined as the percentage of patterns in the test set that are correctly classified by a trained network. Since the performance of Fuzzy ARTMAP depends on the order of pattern presentation in the training set, ten different random orders of pattern presentation were investigated, and performance measures such as the average generalization performance, the worst generalization performance, the best generalization performance, and the standard deviation of the generalization performance were produced for Fuzzy ARTMAP. Other measures of comparison of Ordered Fuzzy ARTMAP and Fuzzy ARTMAP are the sizes of the networks that these two algorithms create, and the numbers of operations required by Ordered Fuzzy ARTMAP as compared to the number of operations required by Fuzzy ARTMAP.

5.3 Comparisons of Ordered Fuzzy ARTMAP and Fuzzy ARTMAP

The only weak link in the procedure that finds an ordered sequence of training patterns for Ordered Fuzzy ARTMAP is that the number of clusters parameter (n_{clust}) must be specified in Stage 2. Our experimental results have shown that a good rule of thumb for choosing the number of clusters n_{clust} , is the *number of classes* or one more than the *number of classes* in the data set. This rule of thumb tends to produce an Ordered Fuzzy ARTMAP with the best

generalization performance.

In Table 1, we show generalization performance comparisons between Fuzzy ARTMAP and Ordered Fuzzy ARTMAP when n_{clust} is chosen to be equal to one more than the *number of classes*. Looking at the results we observe the following: The generalization performance of the Ordered Fuzzy ARTMAP is **better than the worst** generalization performance of Fuzzy ARTMAP by 16.19%, 11.59%, 9.64%, 8.34%, 8.23%, 6.89%, 4.29%, 3.85%, and 1.29% for the Sonar, Glass, Bupa, Iris, Diabetes, Wine, Cars, Balance, and Breast databases, respectively. The generalization performance of the Ordered Fuzzy ARTMAP is **better than the average** generalization performance of Fuzzy ARTMAP by 9.38%, 5.79%, 3.17%, 2.92%, 2.58%, 1.07%, 0.17%, 0.04%, and -0.43% for the Sonar, Glass, Diabetes, Iris, Wine, Cars, Bupa, Breast, and Balance databases, respectively. The generalization performance of the Ordered Fuzzy ARTMAP is **better than the best** generalization performance of Fuzzy ARTMAP by 2.09%, 1.7%, -0.01%, -1.18%, -1.73%, -1.79%, -3.37%, -6.15%, and -7.25% for the Iris, Sonar, Wine, Diabetes, Breast, Cars, Balance, Bupa, and Glass databases, respectively. Negative percentages imply that the corresponding Fuzzy ARTMAP generalization performance (worst, average, or best) is better than the Ordered Fuzzy ARTMAP generalization performance.

In Table 2, we show the size of the network that Ordered Fuzzy ARTMAP created and the average size of the network that Fuzzy ARTMAP created. It is worth pointing out that the size of the neural network architectures that Ordered Fuzzy ARTMAP creates range between 0.80 and 1.52 of the average size of the network architectures that Fuzzy ARTMAP creates. Also, for most of the databases (Sonar, Diabetes, Breast, Bupa, Iris and Glass) the Ordered Fuzzy ARTMAP network size is either smaller or approximately equal to the average Fuzzy ARTMAP network size. In Appendix 3 we perform a very detailed analysis of the number of operations required by the Ordering Algorithm and the average number of operations required by the training phase of Fuzzy ARTMAP. This analysis shows that in both cases the number of operations required is $O(PT)$, where the constant of proportionality in the Ordering Algorithm is approximately equal to n_{clust}^2 , while the constant of proportionality in Fuzzy ARTMAP is approximately equal to $\sum_{e=1}^E n_e$, where n_e is the average number of categories in F_2^a during the e -th epoch of training, and E is the average number of epochs needed by Fuzzy ARTMAP to learn the required task. As can be seen in Table 2, there are

databases (e.g., Diabetes, Bupa, Balance, Cars), where the constant n_{clust}^2 could be a small fraction of $\sum_{e=1}^E n_e$, and as a result the operations required by Ordered Fuzzy ARTMAP could be a small fraction of the operations required by the training phase of Fuzzy ARTMAP. It is worth pointing out that we have repeated the experiments with the above databases for three different collections of training/test sets beyond the ones for which results were reported in Tables 1 and 2 above. The numerical results obtained with these new training/test data sets were slightly different than the ones shown in Tables 1 and 2, but the conclusions obtained from these results were of the same nature as the conclusions derived from Tables 1 and 2.

Since Ordered Fuzzy ARTMAP does not have the on-line capability of Fuzzy ARTMAP, it is fair to compare the performance of Ordered Fuzzy ARTMAP with other classification techniques that have appeared in the literature. In order to make these comparisons we rely on the experimental results produced by Joshi, Ramakrisman, Houstis and Rice [15]. These authors compare a number of neuro-fuzzy, machine learning, and statistical techniques on a variety of databases. The measure of comparison is the generalization performance. The classical machine learning algorithms used in this comparison were: ID3, HOODG, Const, IB, C4.5, Bayes, oneR, Aha-IB, Dis-Bayes, and OC1-Inducer. The statistical techniques used were: Regression models, and Discriminant Analysis. The neuro-fuzzy techniques utilized consisted of: Back-propagation, Back-propagation with Momentum, Quickprop, R-Prop, LVQ1, OLVQ1, LVQ2, and LVQ3. The databases used were: Iris, Pythia, Soybean, Glass, Ionosphere, ECG and Wine. The data were split (as is the the case in this paper) into a training set and a test set (2/3 and 1/3 of the whole dataset, respectively). The reported results correspond to the best set of parameters for each one of the aforementioned techniques [15]. For the Iris database, the range of the generalization performances achieved by the above techniques was from a minimum of 78.5% to a maximum of 95.7%. Ordered Fuzzy ARTMAP achieved a generalization performance of 97.92% without any parameter optimization. For the Wine database, the range of the generalization performances reported by using the above techniques was from a minimum of 90.2% to a maximum of 100%. The 100% performance was achieved only by one algorithm (Simpson's); the rest of the techniques had a best performance of 98%. Ordered Fuzzy ARTMAP achieved a generalization performance of 98.27% without any parameter optimization. For the Glass database the reported generalization performance ranged from a minimum of 83.7% to a maximum of 95.13%. The corresponding Ordered Fuzzy ARTMAP generalization performance was

69.56%. Note that the range of Fuzzy ARTMAP generalization performances for the Glass database was from a minimum of 57.97% to a maximum of 76.81%. Hence, Fuzzy ARTMAP does not perform very well with this database and the poor generalization reported for this database should not necessarily be attributed to any limitations of the proposed Ordering Algorithm. Actually, from [15], we see the best generalization performances (95.13%) attained for the Glass is attributed to Simpson’s algorithm, which is an example of an ART-like architecture; although Simpson’s algorithm was allowed to experiment with parameters in order to optimize generalization performance. The objective in this paper was to propose an Ordering Algorithm that leads to a Fuzzy ARTMAP network with good generalization performance (at least for most databases) without having to resort to excessive experimentation to tune the network’s parameters. We believe that our experimental results, and the above comparisons with other techniques support the validity of our approach.

6 Review – Conclusions

In this paper we introduced a procedure, referred to as the Ordering Algorithm, that identifies a fixed order of training pattern presentation for Fuzzy ARTMAP. The Ordering Algorithm is based on the Max-Min clustering algorithm. The combination of the Ordering Algorithm and Fuzzy ARTMAP is called Ordered Fuzzy ARTMAP. Experiments with nine different classification problems have shown that Ordered Fuzzy ARTMAP attains a superior generalization performance as compared to the average performance of Fuzzy ARTMAP, and in certain cases as good as, or better than the best Fuzzy ARTMAP generalization performance. The average and best generalization performances are obtained over a fixed number of experiments with Fuzzy ARTMAP corresponding to different orders of training pattern presentations. We also demonstrated that under mild conditions on the pattern classification tasks, the operations required by the Ordering Algorithm is a fraction of the operations required by the training phase of Fuzzy ARTMAP for a single order of training pattern presentation. Furthermore, the sizes of the network architectures that Ordered Fuzzy ARTMAP creates are comparable to the average size of the network architectures that Fuzzy ARTMAP creates. Finally, the proposed procedure for ordering the training data may also be applied to other ART-type architectures, such as Simpson’s Min-Max architecture [1], the LAPART architecture of Healy et. al. [16],

and the ARTEMAPQ and ARTMAP-IC architectures of Carpenter's et. al. [17].

Appendix 1

In order to explain the choice of the first pattern in the Ordering Algorithm, let us consider Example 2 of Figure 4. Example 2 was fully described in Section 4. In this example, the data $\{0, 0.05, 0.1, 0.4, 0.45, 0.51\}$ belong to Class 1, while the data $\{0.2, 0.25, 0.3, 0.6, 0.65, 0.7\}$ belong to Class 2. If we apply equation (3) to these datapoints: Datum “0” with representation $(a_1, a_2) = (0, 1)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |1 - 0| = 1$, Datum “0.05” with representation $(a_1, a_2) = (0.05, 0.95)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.95 - 0.05| = 0.9$, Datum “0.1” with representation $(a_1, a_2) = (0.1, 0.9)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.9 - 0.1| = 0.8$, Datum “0.2” with representation $(a_1, a_2) = (0.2, 0.8)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.8 - 0.2| = 0.6$, Datum “0.25” of with representation $(a_1, a_2) = (0.25, 0.75)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.75 - 0.25| = 0.5$, Datum “0.3” with representation $(a_1, a_2) = (0.3, 0.7)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.7 - 0.3| = 0.4$, Datum “0.4” with representation $(a_1, a_2) = (0.4, 0.6)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.6 - 0.4| = 0.2$, Datum “0.45” with representation $(a_1, a_2) = (0.45, 0.55)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.55 - 0.45| = 0.1$, Datum “0.51” of with representation $(a_1, a_2) = (0.51, 0.49)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.49 - 0.51| = 0.02$, Datum “0.6” with representation $(a_1, a_2) = (0.6, 0.4)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.4 - 0.6| = 0.2$, Datum “0.65” with representation $(a_1, a_2) = (0.65, 0.35)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.35 - 0.65| = 0.3$, Datum “0.7” with representation $(a_1, a_2) = (0.7, 0.3)$ yields $\sum_{i=1}^1 |a_{i+1} - a_i| = |0.3 - 0.7| = 0.4$. From the above calculations it is easy to see that the datum that maximizes $\sum_{i=1}^{M_a} |a_{i+M_a} - a_i|$ is datum “0”, one of the endpoints of the interval $[0, 1]$. In higher dimensions, we expect Stage 1 to give us a datapoint that is close to one of the vertices of the hypercube in which the input patterns reside.

Appendix 2

Theorem 1:

The order sequence of the training input patterns that the Ordering Algorithm produces is independent of any permutations of their indices.

Proof: We can prove Theorem 1 in the general case, when dealing with an arbitrary number of training input patterns and for an arbitrary choice of n_{clust} , but have not provided this proof due to the cumbersome notation involved. We believe that presenting an example which verifies the theorem is more enlightening. Let us, therefore, refer to Example 1 of Figure 3 (see Section 4 in the main text). We will consider two distinct permutations of the training input patterns. Permutation 1 assigns indices in increasing order, first to input patterns of Class 1, and then to input patterns of Class 2. Within a class indices are assigned in increasing order, according to increasing distance from the vertex “0”. Hence, for Permutation 1 we designate the collection of input patterns as follows:

$$\begin{aligned}
 \mathbf{I}^1 &= (0.0, 1.0) \\
 \mathbf{I}^2 &= (0.1, 0.9) \\
 \mathbf{I}^3 &= (0.4, 0.6) \\
 \mathbf{I}^4 &= (0.5, 0.5) \\
 \mathbf{I}^5 &= (0.2, 0.8) \\
 \mathbf{I}^6 &= (0.3, 0.7) \\
 \mathbf{I}^7 &= (1.0, 0.0)
 \end{aligned} \tag{5}$$

Permutation 2 assigns indices in increasing order first to input patterns of Class 2, and then to input patterns, of Class 1. Within a class indices are assigned in increasing order, according to increasing distance from the vertex “0”.

Hence, for Permutation 2 we designate the collection of input patterns as follows:

$$\begin{aligned}
 \hat{\mathbf{I}}^1 &= (0.2, 0.8) \\
 \hat{\mathbf{I}}^2 &= (0.3, 0.7) \\
 \hat{\mathbf{I}}^3 &= (1.0, 0.0) \\
 \hat{\mathbf{I}}^4 &= (0.0, 1.0) \\
 \hat{\mathbf{I}}^5 &= (0.1, 0.9) \\
 \hat{\mathbf{I}}^6 &= (0.4, 0.6) \\
 \hat{\mathbf{I}}^7 &= (0.5, 0.5)
 \end{aligned} \tag{6}$$

We denote by S_T the set of training input patterns of Permutation 1 (initially taken to be all the \mathbf{I} ’s), and by \hat{S}_T the set of training input patterns of Permutation 2 (initially taken to be all the $\hat{\mathbf{I}}$ ’s). The application of equation (3) of Stage 1 of the Ordering Algorithm to the input patterns, designated in Permutation 1, produces the following

numbers in order of increasing index: 1.0, 0.8, 0.2, 0.0, 0.6, 0.4, 1.0. The maximum number (1.0) corresponds to patterns $\mathbf{I}^1 = (0.0, 1.0)$ and $\mathbf{I}^7 = (1.0, 0.0)$; we choose (arbitrarily) pattern $\mathbf{I}^1 = (0.0, 1.0)$ since it is the input pattern closest to vertex “0”. Pattern $\mathbf{I}^1 = (0.0, 1.0)$ is eliminated from the training set S_T . The application of equation (3) of Stage 1 of the Ordering Algorithm to the input patterns, designated in Permutation 2, produces the following numbers in order of increasing index: 0.6, 0.4, 1.0, 1.0, 0.8, 0.2, 0.0. The maximum number (1.0) corresponds to patterns $\hat{\mathbf{I}}^3 = (1.0, 0.0)$ and $\hat{\mathbf{I}}^4 = (0.0, 1.0)$; we choose (arbitrarily) pattern $\hat{\mathbf{I}}^4 = (0.0, 1.0)$ since it is the input pattern closest to vertex “0”. Pattern $\hat{\mathbf{I}}^4 = (0.0, 1.0)$ is eliminated from the training set \hat{S}_T . Thus, for both permutations, the application of Stage 1 of the Ordering Algorithm led us to choose the same input pattern to be presented first to Ordered Fuzzy ARTMAP.

The application of equation (4) of Stage 2 of the Ordering Algorithm to the input patterns, designated in Permutation 1, produces the following distances in order of increasing index (remember that the pattern with index 1 in Permutation 1 was eliminated from S_T in Stage 1): 0.1, 0.4, 0.5, 0.2, 0.3, 1.0. The maximum number (1.0) corresponds to pattern $\mathbf{I}^7 = (1.0, 0.0)$, and as result this is the pattern chosen to be presented next to Ordered Fuzzy ARTMAP. Pattern $\mathbf{I}^7 = (1.0, 0.0)$ is eliminated from the training set S_T . The application of equation (4) of Stage 2 of the Ordering Algorithm to the input patterns, designated in Permutation 2, produces the following distances in order of increasing index (remember that the pattern with index 4 in Permutation 2 was eliminated from \hat{S}_T in Stage 1): 0.2, 0.3, 1.0, 0.1, 0.4, 0.5. The maximum number (1.0) corresponds to pattern $\hat{\mathbf{I}}^3 = (1.0, 0.0)$, and as result this is the pattern chosen to be presented next to Ordered Fuzzy ARTMAP. Pattern $\hat{\mathbf{I}}^3 = (1.0, 0.0)$ is eliminated from the training set \hat{S}_T . Since n_{clust} for this example equals 2, Stage 2 is completed at this point.

The application of Stage 3 of the Ordering Algorithm to the input patterns designated in Permutation 1 produces distances 0.1, 0.4, 0.5, 0.2, 0.3 from the first cluster point, and distances 0.9, 0.6, 0.5, 0.8, 0.7 from the second cluster point; distances are reported in order of increasing index for the Permutation 1 patterns. Hence, according to the Stage 3 rules, the remaining input patterns will be presented in the order $\mathbf{I}^2 = (0.1, 0.9)$, $\mathbf{I}^5 = (0.2, 0.8)$, $\mathbf{I}^6 = (0.3, 0.7)$, $\mathbf{I}^3 = (0.4, 0.6)$, and $\mathbf{I}^4 = (0.5, 0.5)$. The application of Stage 3 of the Ordering Algorithm to the input patterns designated in Permutation 2 produces distances 0.2, 0.3, 0.1, 0.4, 0.5 from the first cluster point, and

distances 0.8, 0.7, 0.9, 0.6, 0.5 from the second cluster point, where distances are reported in order of increasing index for the Permutation 2 patterns. According to the Stage 3 rules, the remaining input patterns will be presented in the order $\hat{\mathbf{I}}^5 = (0.1, 0.9)$, $\hat{\mathbf{I}}^1 = (0.2, 0.8)$, $\hat{\mathbf{I}}^2 = (0.3, 0.7)$, $\hat{\mathbf{I}}^6 = (0.4, 0.6)$, and $\hat{\mathbf{I}}^7 = (0.5, 0.5)$. Consequently, both Permutations 1 and 2 lead to the same ordering of the training input patterns, a result which verifies the validity of Theorem 1.

Appendix 3

The major operations involved in the computations associated with the Ordering Algorithm are: addition, subtraction, multiplication, absolute value, and comparison. We assume these operations are equally expensive. This is in fact a reasonable assumption for CISC (complex instruction set) computers. The operations required by the Ordering Algorithm will be separated into two categories: Operations that are required to calculate the necessary distances, and operations required to find the minimum or maximum of these distances.

Operations for the Ordering Algorithm required to calculate distances

Stage 1: $3M_a PT$ operations to calculate $\sum_{i=1}^{M_a} |a_{i+M_a} - a_i|$ for the PT training input patterns. Each pattern requires M_a subtractions, M_a absolute value operations and M_a additions.

Stage 2: For the second cluster center we need $3M_a(PT - 1)$ calculations to compute the distances of $(PT - 1)$ training input patterns from the first cluster center. The per pattern operations are $3M_a$ because in order to compute the Euclidean distance of two patterns of dimensionality M_a we need M_a subtractions, M_a multiplications and M_a additions. For the third cluster center we need $3M_a 2(PT - 2)$ calculations to compute the distances of $(PT - 2)$ training input patterns from cluster centers 1 and 2. Eventually, for the n_{clust} cluster centers we need $3M_a(n_{clust} - 1)(PT + 1 - n_{clust})$ operations to compute the distances of $(PT + 1 - n_{clust})$ training input patterns from $(n_{clust} - 1)$ cluster centers. Hence, overall we need

$$3M_a[(PT - 1) + 2(PT - 2) + \dots + (n_{clust} - 1)(PT + 1 - n_{clust})] \quad (7)$$

operations to calculate the necessary distances in Stage 2 of the Ordering Algorithm.

Stage 3: To calculate the distances of $(PT - n_{clust})$ training input patterns from n_{clust} cluster centers we need $3M_a n_{clust}(PT - n_{clust})$ operations. Combining the distance-related operations for Stages 1, 2, and 3 of the Ordering Algorithm we see that we need

$$3M_a[PT + (PT - 1) + 2(PT - 2) + \dots + n_{clust}(PT - n_{clust})] \quad (8)$$

operations, or approximately (if $n_{clust} \ll PT$)

$$3M_a PT[1 + 1 + 2 + \dots + n_{clust}] = 3M_a PT[1 + n_{clust}(1 + n_{clust})/2] \quad (9)$$

operations.

Operations for the Ordering Algorithm required to calculate max and/or min of distances

Stage 1: No maximum or minimum distance calculations are required in this stage.

Stage 2: For the second cluster center we need $(PT - 2)$ operations to find the minimum of $(PT - 1)$ distances. For the third cluster center we need $2(PT - 3)$ operations (comparisons) to find the minimum of $(PT - 2)$ distances from cluster center 1 and the minimum of $(PT - 2)$ distances from cluster center 2. Then, we need one operation to find the maximum of these two minimum distances. Eventually for the n_{clust} -th cluster center we need $(n_{clust} - 1)(PT - n_{clust})$ to find the minimum of $(PT + 1 - n_{clust})$ distances from cluster center 1, the minimum of $(PT + 1 - n_{clust})$ distances from cluster center 2, and eventually the minimum of $(PT + 1 - n_{clust})$ distances from cluster center $n_{clust} - 1$. Then, we need $(n_{clust} - 2)$ operations to find the maximum of these $(n_{clust} - 1)$ minimum distances. Hence, overall we need

$$[(PT - 2) + 2(PT - 3) + \dots + (n_{clust} - 1)(PT - n_{clust})] + [0 + 1 + \dots + (n_{clust} - 2)] \quad (10)$$

operations, or approximately (if $n_{clust} \ll PT$)

$$PT[1 + 2 + \dots + (n_{clust} - 1)] = PT[n_{clust}(n_{clust} - 1)/2] \quad (11)$$

operations to find the *max-min* of the necessary distances in Stage 2 of the Ordering Algorithm.

Stage 3: In Stage 3 we need to find the minimum of n_{clust} distances of each one of the remaining $(PT - n_{clust})$ training input patterns from the n_{clust} cluster centers. This requires

$$(PT - n_{clust})(n_{clust} - 1) \quad (12)$$

operations, or approximately (if $n_{clust} \ll PT$)

$$PT n_{clust} \quad (13)$$

operations. Then, we need to sort these $(PT - n_{clust})$ minimum distances. We know that these distances range in the interval $[0, M_a]$. We first make these distances integers by multiplying each one of them by an appropriate integer N (e.g., $N = 10$). This puts the distances in the range $[0, M_a N]$. Subsequently we sort these integers using radix sorting ([18]). This procedure consists of the following steps.

1. Initialize $M_a N$ empty queues, one for each integer in the range 1 to $M_a N$.
2. Scan the sequence of integer distances $d_1, d_2, \dots, d_{PT-n_{clust}}$ from left to right, placing element d_i in the d_i queue.
3. Concatenate the queues with nonzero contents to obtain the sorted sequence.

Multiplying the $(PT - n_{clust})$ distances by N requires $(PT - n_{clust})$ operations. We assume that it takes one operation to insert an element into the i -th queue. So to place $(PT - n_{clust})$ elements we need $(PT - n_{clust})$ operations. Concatenating the $N M_a$ queues requires $N M_a$ operations. Hence, for the sorting in Stage 3, we need

$$2(PT - n_{clust}) + M_a N \quad (14)$$

operations, or approximately (for $n_{clust} \ll PT$)

$$2PT + M_a N \quad (15)$$

operations. Combining the max-min, and sorting related operations for Stages 1, 2, and 3 of the Ordering Algorithm we see that we need

$$PT[n_{clust}(n_{clust} - 1)/2] + PT n_{clust} + 2PT + M_a N = PT[n_{clust}(n_{clust} + 1)/2 + 2 + M_a N/PT] \quad (16)$$

operations. The major operations involved in the computations associated with the training phase of Fuzzy ARTMAP are: addition, minimum operation, division, and comparisons. We assume that all these operations have the same cost. For the training phase of Fuzzy ARTMAP we can also break down the required operations into distance related operations (which correspond to calculations of the bottom-up inputs to all committed nodes, plus one uncommitted node, in F_2^a), and operations related to finding the maximum of these distances. Furthermore, for Fuzzy ARTMAP

we have to consider the operations required to calculate weight changes as well; for simplicity of presentation we lump these weight-change related operations with the distance related equations.

Distance related operations for Fuzzy ARTMAP

Let us assume that the average number of nodes for which bottom-up inputs need to be computed in epoch 1 of Fuzzy ARTMAP training is equal to n_1 . Recall that the bottom-up input to node j in F_2^a of Fuzzy ARTMAP is given by

$$T_j^a = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\beta_a + |\mathbf{w}_j|} \quad (17)$$

where \wedge denotes the minimum operator applied to two vectors. The minimum operator applied on two vectors \mathbf{x} and \mathbf{y} is a vector \mathbf{z} with components the minimum of the corresponding components of \mathbf{x} and \mathbf{y} . Also the notation $|\mathbf{x}|$ stands for the size of a vector \mathbf{x} , and the size of a vector is defined to be the sum of its components. Hence, the calculation of a single bottom-up input requires $2M_a$ minimum operations, and $4M_a$ addition operations (for simplicity we are omitting the single division operation). Since we have assumed an average of n_1 nodes in F_2^a in the first epoch of training, we need, on the average,

$$6M_a n_1 PT \quad (18)$$

operations for the calculation of the bottom-up inputs only. For the change of the weights during every pattern presentation we need $2M_a$ operations. Hence, the total number of operations for weight changes during an epoch is equal to $2M_a PT$. Thus for the first epoch average number of operations needed by Fuzzy ARTMAP equals

$$3M_a PT(2n_1 + 2/3) \quad (19)$$

Assume that the average number of epochs required by Fuzzy ARTMAP to converge is E . For epochs beyond epoch 1, similar formulas are valid for the number of operations required but the average number of categories in F_2^a changes to n_2 for epoch 2, n_3 for epoch 3, and eventually n_E for epoch E . Note that $n_E \geq \dots \geq n_3 \geq n_2 \geq n_1$. Hence, the total average number of operations needed for distance related and weight changes calculations, until Fuzzy ARTMAP converges, is equal to

$$3M_a PT(2/3E + 2 \sum_{e=1}^E n_e) \quad (20)$$

Operations in Fuzzy ARTMAP related to calculating maximum of distances

In the first epoch of training in Fuzzy ARTMAP we need to find the maximum of n_1 distances (the T 's) for every pattern presentation. Hence, the average number of operations required to find these maximum distances in the first epoch of training is equal to

$$PT(n_1 - 1) \quad (21)$$

Similarly we can find the number of operations required to obtain the maximum of n_2 distances (PT times) in epoch 2, the maximum of n_3 distances (PT times) in epoch 3, and eventually the maximum of n_E distances (PT times) in epoch E . Overall the average number of maximum distance related operations in the training phase of Fuzzy ARTMAP is equal to

$$PT \left[\sum_{e=1}^E n_e - E \right] \quad (22)$$

Note that in equations (21) through (24) we have omitted the operations required when resets of nodes in F_2^a occur.

Cumulative operations for the Ordering Algorithm and Fuzzy ARTMAP

Combining the **number of operations required by the Ordering Algorithm** we get

$$3M_a PT[1 + n_{clust}(1 + n_{clust})/2] + PT[n_{clust}(n_{clust} + 1)/2 + 2 + M_a N/PT] \quad (23)$$

Also, combining the **average number of operations required by Fuzzy ARTMAP** we get

$$3M_a PT(2/3E + 2 \sum_{e=1}^E n_e) + PT[\sum_{e=1}^E n_e - E] \quad (24)$$

Assuming that $M_a N/PT = O(1)$, we see from the above equations that the number of operations required by the Ordering Algorithm and the average number of operations required by Fuzzy ARTMAP are $O(PT)$. As a result, a

more accurate comparison between these two algorithms should rely on the actual values of n_{clust}^2 and $\sum_{e=1}^E n_e$ (see main text, second paragraph of Section 5.3, for more details).

References

- [1] P. K. Simpson, "Fuzzy min-max neural networks – Part 1: Classification," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 776-786, September 1992.
- [2] 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 Transactions on Neural Networks*, Vol. 3, No. 5, pp. 698-713, September 1992.
- [3] S. Grossberg, "Adaptive pattern recognition and universal recoding II: Feedback, expectation, olfaction, and illusions," *Biological Cybernetics*, Vol. 23, pp. 187-202, 1976.
- [4] D. Rumelhart, and J. McClelland, Eds. *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*. MA: MIT Press, 1986.
- [5] J. T. Tou and R. C. Gonzalez. Pattern Recognition Principles. Addison Wesley Publishing Company, Reading, MA, 1976.
- [6] 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, pp. 759-771, 1991.
- [7] M. Georgiopoulos, H. Fernlund, G. Bebis, and G. L. Heileman, "Order of search in Fuzzy ART and Fuzzy ARTMAP: Effect of the choice parameter," *Neural Networks*, Vol. 9, No. 9, pp. 1541-1559, 1996.
- [8] P. Murphy, and D. Ada, UCI Repository of Machine Learning databases, *Technical Report*, <http://www.ics.edu/~mlearn/MLRepository.html>, Department of Computer Science, University of California, Irvine, CA, 1994.
- [9] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annual Eugenics*, 7, Part II, pp. 179-188, 1936.
- [10] S. Aeberhard, D. Coomans, and O. De Vel, "Comparison of classifiers in high dimensional settings," *Technical Report, No. 92-02, Dept. Of Mathematics and Statistics, James Cook University of North Queensland*, 1992.
- [11] R. P. Gorman, and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, Vol. 1, pp. 75-89, 1988.
- [12] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes melitus," *Proceedings of the Symposium on Computer Applications and Medical Care*, pp. 261-265, IEEE Computer Society Press, 1990.
- [13] O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, Vol. 23, No. 5, pp. 1-18, September 1990.
- [14] R. S. Siegler, "Three aspects of cognitive development," *Cognitive Psychology*, Vol. 8, pp. 481-520, 1976.
- [15] A. Joshi, N. Ramakrishman, E. N. Houstis, and J. R. Rice, "On neurobiological, neuro-fuzzy, machine learning, and statistical pattern recognition techniques," *IEEE Transactions on Neural Networks*, Vol. 8, No. 1, pp. 18-31, 1997.
- [16] M. J. Healy, T. P. Caudell, and S. D. G. Smith, "A neural network architecture for pattern sequence verification through inferencing," *IEEE Transactions on Neural Networks*, Vol. 4, No. 1, pp. 9-20, 1993.
- [17] G. A. Carpenter, and N. Markuzon, "ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases," *Neural Networks*, Vol. 11, pp. 323-336, 1998.
- [18] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company, Reading, MA, 1974.

Table 1: Generalization performances of the Fuzzy ARTMAP and the Ordered Fuzzy ARTMAP with $n_{clust} = (\text{number of classes}) + 1$

Database	Fuzzy ARTMAP				Ordered Fuzzy ARTMAP	
	Worst Gen.	Best Gen.	avg.Gen.	std.dev.	n_{clust}	Gen.
Sonar	63.77	78.26	70.58	4.15	3	79.96
Diabetes	61.57	70.98	66.63	2.57	3	69.80
Breast	93.10	96.12	94.35	0.95	3	94.39
Bupa	47.37	63.16	56.84	4.22	3	57.01
Iris	89.58	95.83	95.00	1.91	4	97.92
Wine	91.38	98.28	95.69	2.70	4	98.27
Balance	71.63	78.85	75.91	2.42	4	75.48
Cars	63.21	69.29	66.43	2.13	5	67.50
Glass	57.97	76.81	63.77	6.18	7	69.56

Table 2: Average network size for Fuzzy ARTMAP and network size of Ordered Fuzzy ARTMAP with $n_{clust} = (\text{number of classes}) + 1$

Database	Fuzzy ARTMAP	Ordered Fuzzy ARTMAP	
	Avg. Net Size	n_{clust}	Net Size
Sonar	6	3	5
Diabetes	43	3	44
Breast	8	3	9
Bupa	31	3	31
Iris	5	4	4
Wine	4	4	6
Balance	79	3	120
Cars	46	4	56
Glass	27	7	30

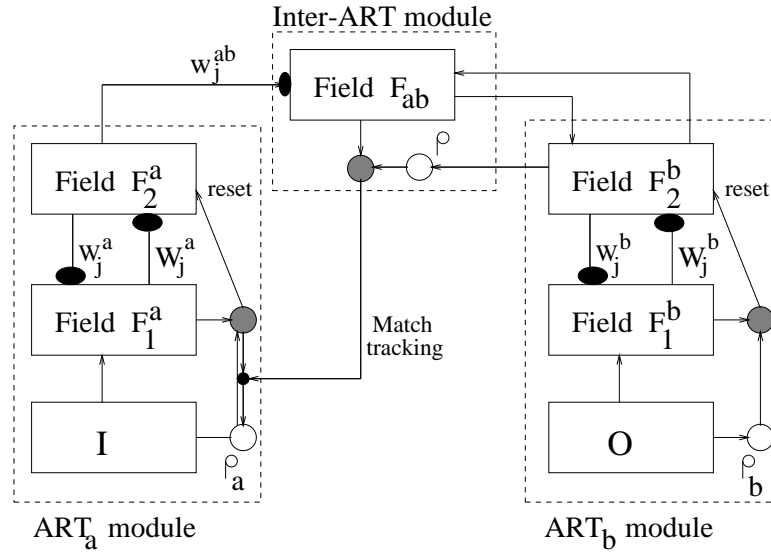


Figure 1: A block diagram of the Fuzzy ARTMAP neural network architecture.

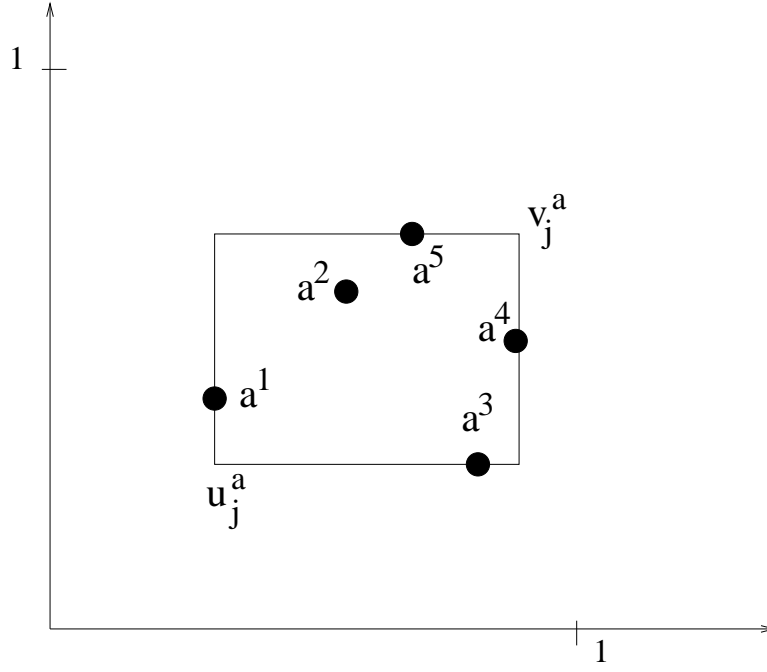


Figure 2: The hyperrectangle which has coded the patterns $\mathbf{I}^1 = (\mathbf{a}^1, (\mathbf{a}^1)^c)$, $\mathbf{I}^2 = (\mathbf{a}^2, (\mathbf{a}^2)^c)$, $\mathbf{I}^3 = (\mathbf{a}^3, (\mathbf{a}^3)^c)$, $\mathbf{I}^4 = (\mathbf{a}^4, (\mathbf{a}^4)^c)$, and $\mathbf{I}^5 = (\mathbf{a}^5, (\mathbf{a}^5)^c)$.

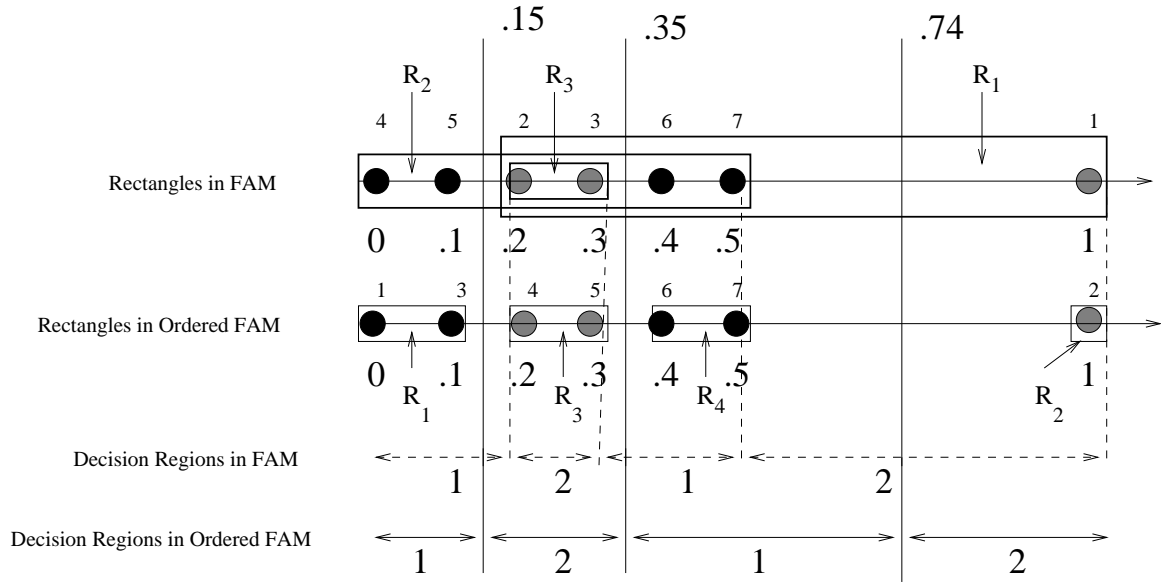


Figure 3: Rectangles and decision Regions of Fuzzy ARTMAP (FAM) and Ordered Fuzzy ARTMAP (Ordered FAM) for Example 1.

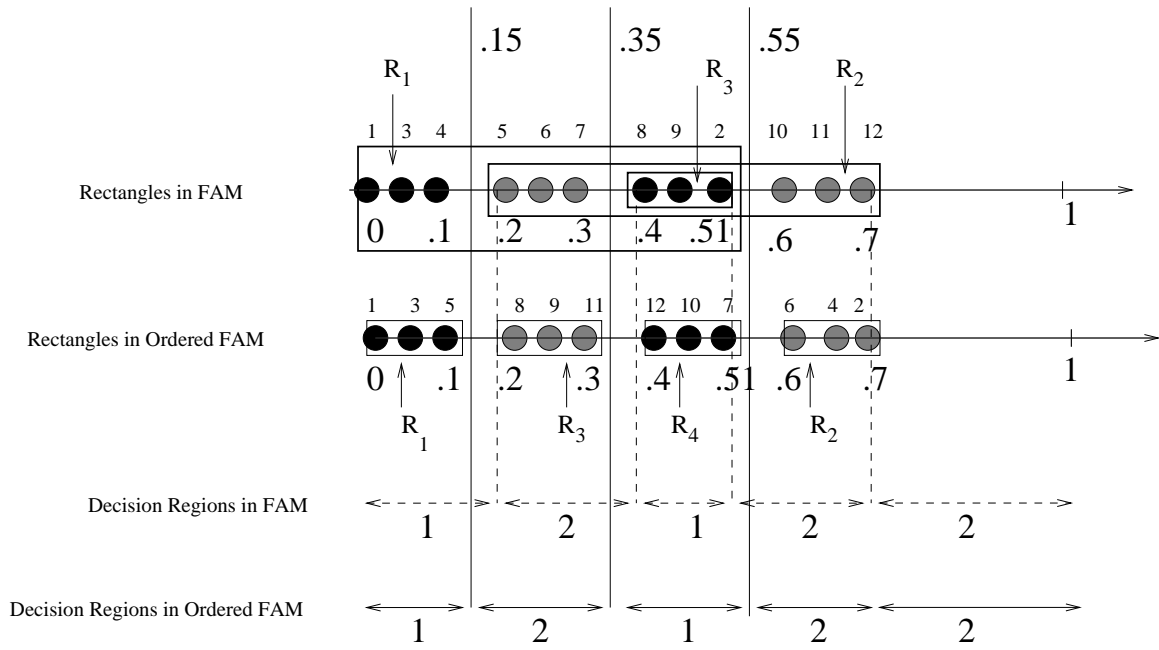


Figure 4: Rectangles and decision Regions of Fuzzy ARTMAP (FAM) and Ordered Fuzzy ARTMAP (Ordered FAM) for Example 2.