# PIPELINING OF FUZZY ARTMAP (FAM) WITHOUT MATCH TRACKING

JOSE CASTRO(*), JIMMY SECRETAN(**), MICHAEL GEORGIOPOULOS(**),
RONALD F. DEMARA (*), GEORGIOS ANAGNOSTOPOULOS(***),
AVELINO GONZALEZ (**)
(*) Comp Eng., Instituto Tecnológico de Costa Rica, Cartago, Costa Rica
(**) Dept. of ECE, University of Central Florida, Orlando, FL 32816
(***) Dept. of ECE, Florida Institute of Technology, Melbourne, FL 32901

***ABSTRACT***

Fuzzy ARTMAP (FAM) is a neural network architecture that can establish the correct mapping between real-valued input patterns and correct labels in a variety of classification problems. Nevertheless, as the size of the dataset grows to thousands and hundreds of thousands, FAM's convergence time slows down considerably. In this paper we focus on a FAM variant called no-match tracking FAM (NMT-FAM). We propose a coarse grain parallelization of the NMT-FAM, based on a pipeline, and show that the parallelization strategy achieves linear speed-up in the order of *p* (number of processors). Experiments on the Covertype database support our results. We have also shown, but not included in this paper, that the parallelized NMT-FAM is equivalent to the sequential NMT-FAM, it also possesses a number of good properties. Our work in this paper is an effort in the direction of demonstrating that FAM can, through appropriate parallelization strategies, be used to mine data from large databases.

## 1. INTRODUCTION

Neural network algorithms have prohibitively slow training times, especially when they learn from large databases. Even Fuzzy ARTMAP (Carpenter, 1992), one of the fastest neural network algorithms in terms of training time, tends to exhibit slow convergence time as the size of the network increases. One way to address this problem is extensive use of parallelization. One of the works related to the parallelization of ART that is worth mentioning is the work by Manolakos (1998), where he has implemented a non-supervised ART neural network (ART1) on a ring of processors. Fuzzy ARTMAP has many desirable characteristics, such as on-line learning capabilities, fast training time, the ability to solve any mapping (classification) problem, an ease to provide explanations for the answers that it produces, etc. But Fuzzy ARTMAP's potential is compromised by the fact that its convergence speed to a solution, decreases considerably for problems with large datasets. This paper addresses this issue, the issue of Fuzzy ARTMAP parallelization so that it becomes more efficient, when confronted with the training of large datasets.

## 2. FAM AND NO MATCH-TRACKING FAM ALGORITHMS

The Fuzzy ARTMAP architecture consists of three layers or fields of nodes (see Figure 1). It is assumed here that the reader is familiar with the Fuzzy ARTMAP layers, their functionality, and the associated Fuzzy ARTMAP weights. It is also assumed here that the reader is familiar with the two network parameters in FAM, the choice parameter $\beta_a$, and the baseline vigilance parameter $\overline{\rho}_a$. The terminology that we have adopted is to designate inputs by **I**, outputs by **O**, template weights by $\mathbf{w}_j^a$ and inter-ART

weights by $\mathbf{W}_j^{ab}$. Fuzzy ARTMAP can operate in two distinct phases: the *training phase* and the *performance phase*. In the training phase, Fuzzy ARTMAP is presented with a sequence of input/output pairs and it goes through a set of designated operations (to be described below) to learn the correct mapping from inputs to desired outputs. In this paper, only the on-line training phase of Fuzzy ARTMAP is considered, where an input output pair from the training set is only presented once. In the performance phase the trained Fuzzy ARTMAP is presented with an input pattern and it is required to produce a response (predicted output pattern). In this phase Fuzzy ARTMAP is usually presented with a list of input patterns whose correct output (response) is known; this way the trained Fuzzy ARTMAP's performance can be assessed. Prior to initiating the training phase of Fuzzy ARTMAP the top–down weights (the $w_{ji}^a$'s) are chosen equal to 1. There are three major operations that take place during the presentation of a training input/output pair (e.g., $\left(\mathbf{I}^r, \mathbf{O}^r\right)$) to Fuzzy ARTMAP. One of the specific operands involved in all of these operations is the *fuzzy min operand*, designated by the symbol ^. Actually, the fuzzy min operation of two vectors **x**, and **y**, designated as **x**^**y**, is a vector whose components are equal to the minimum of components of **x** and **y**. Another specific operand involved in these equations is designated by the symbol $\left|\bullet\right|$. In particular, |**x**| is the size of a vector **x** and is defined to be the sum of its components.

**Operation 1:** Calculation of bottom up inputs to every node *j* in $F_2^a$, as follows:

$$T_j^a = \frac{\left|\mathbf{I}^r \wedge \mathbf{w}_j^a\right|}{\left|\mathbf{w}_j^a\right| + \beta_a} \tag{1}$$

after calculation of the bottom up inputs the node $j_{\max}$ with the maximum bottom up input is chosen.

**Operation 2:** The node $j_{\max}$ with the maximum bottom up input is examined to determine whether it passes the vigilance criterion. A node passes the vigilance criterion if the following condition is met:

$$\frac{\left|\mathbf{I}^r \wedge \mathbf{w}_j^a\right|}{\left|\mathbf{I}^r\right|} \geq \rho_a \tag{2}$$

if the vigilance criterion is satisfied we proceed with operation 3 otherwise node $j_{\max}$ is disqualified and we find the next node in sequence in $F_2{}^a$ that maximizes the bottom up input. Eventually we will end up with a node $j_{\max}$ that maximizes the bottom up input and passes the vigilance criterion.

**Operation 3:** This operation is implemented only after we have found a node $j_{\max}$ that maximizes the bottom-up input of the remaining nodes in competition and that passes the vigilance criterion. Operation 3 determines whether this node $j_{\max}$ passes the prediction test. The prediction test checks if the inter--ART weight vector emanating from node $j_{\max}$

$$\text{ie. } \mathbf{W}_{j_{\max}}^{ab} = \left(W_{j_{\max}1}^{ab}, W_{j_{\max}2}^{ab}, \cdots, W_{j_{\max},N_b}^{ab}\right) \tag{3}$$

matches exactly the desired output vector $\mathbf{O}^r$ (if it does this is referred to as *passing the prediction test*). If the node does *not* pass the prediction test, the vigilance parameter $\rho_a$ is increased to the level of

$$\rho_a \leftarrow \frac{\left|\mathbf{I}^r \wedge \mathbf{w}^a_{j\max}\right|}{\left|\mathbf{I}^r\right|} + \varepsilon \tag{4}$$

where $\varepsilon$ is a very small number. Then, node $j_{\max}$ is disqualified, and the next in sequence node that maximizes the bottom-up input and passes the vigilance is chosen. The above operation of FAM is called match-tracking operation. If, on the other hand, node $j_{\max}$ passes the predictability test, the weights in Fuzzy ARTMAP are modified as follows:

$$\mathbf{w}^a_{j_{\max}} \leftarrow \mathbf{w}^a_{j_{\max}} \wedge \mathbf{I}^r, \mathbf{W}^{ab}_{j_{\max}} \leftarrow \mathbf{O}^r \tag{5}$$

A modification of Fuzzy ARTMAP that is amenable to parallelization is the "no match-tracking" Fuzzy ARTMAP. This modification was proposed by Anagnostopoulos (2003) and it was shown that it can actually improve the performance of Fuzzy ARTMAP on some databases. The no-match tracking Fuzzy ARTMAP algorithm is the same as Fuzzy ARTMAP but it bypasses the match-tracking operation (described above). That is if a node that maximizes the bottom-up input and satisfies the vigilance is selected and this node does not pass the prediction test a new node (uncommitted node) in the category representation layer of Fuzzy ARTMAP is activated.

## 3. PARALLEL, NO-MATCH TRACKING FAM IMPLEMENTATION

Anagnostopoulos's FAM variant is particular amenable to production-line style pipeline parallel implementation since patterns can be evenly distributed among the nodes in the pipeline. The elimination of match-tracking makes the learning of the pattern a one-pass over the pipeline procedure, and different patterns can be processed on the different pipeline steps to achieve optimum parallelization. For the implementation of the no-match tracking FAM we first introduce a number of definitions. The algorithm itself (parallel, no-match tracking FAM implementation) is shown after the definitions are introduced. In the description of the parallel no-match tracking FAM the initialization procedure *(INIT(p))* and *WINNER* are not described due to lack of space. More details about these procedures as well as the algorithm presented here can be found in (Castro, 2004).

$n$: number of processors in the pipeline

$k$: index of current process, $k \in \{0,1,\ldots,n-1\}$

$p$: packet size, number of patterns sent downstream, 2p = number of templates sent upstream

$\mathbf{I}^i$: input pattern $i$ of the current packet in the pipeline. $i \in \{1,2,\ldots,p\}$.

$w^i$: current best candidate template for input pattern $\mathbf{I}^i$.

$T^i$: current maximum activation for input pattern $\mathbf{I}^i$.

*myTemplates*: :set of templates that belong to the current processor.

*nodes*: variable local to the current processor that holds the total number of templates the process is aware of (its own plus the templates of other processors)

*myShare*: amount of templates that the current process should have.

$w^i_{k-1}$: template $i$ coming from previous process in the pipeline.

$w^i_{k+1}$: template $i$ coming from next process in the ring.

$w^i$: template $i$ going to next process in the ring.

$w^i_{to(k-1)}$ : template $i$ going to previous process in the pipeline.

$\mathbf{I}.class$ : class label associated with a given input pattern.

$w.class$ : class label associated with a given input template.

$index(w)$ : sequential index assigned to the template.

$newNodes$ : number of created nodes on a given iteration to communicate upstream in the pipeline.

$newNodes_{k+1}$ : number of created nodes on a given iteration communicated from processor k+1 in the pipeline.

Process $(k, n, \overline{\rho}_a, \beta_a, p)$

```
1   INIT (p)
2   while continue
3   do
4       while |myTemplates| > myShare
5       do
6         EXTRACT-TEMPLATE (myTemplates, {w^i_{to(k-1)}})
7       SEND-NEXT (k, n, {(w^i, I^i, T^i): i = 1, ..., p})
8         RECV-NEXT (k, n, {w^i_{k+1} : i = 1, ..., 2p}, newNodes_{k+1})
9       SEND-NEXT (k, {w^i_{to(k-1)} : i = 1, ..., 2p}, newNodes)
10        RECV-NEXT (k, {(w^i_{(k-1)}, I^i_{k-1}, T^i_{k-1}): i = 1, ..., p})
11      newNodes ← newNodes_{k+1}
12      S ← {w^i_{k+1}}
13        for each i in {1,2, ..., p}
14        do WINNER (I^i, w^i, T^i, ρ̄_a, β_a, S)
15          myTemplates ← myTemplates ∪ S
16        if I^i_{k-1} = EOF
17            then continue ← FALSE
18            else S ← {w^i_{to(k-1)}}
19                 for each i in {1,2, ..., p}
20                 do WINNER (I^i_{k-1}, w^i_{k-1}, T^i_{k-1}, ρ_a, β_a, S)
21                      (I^i, w^i, T^i) ← (I^i_{k-1}, w^i_{k-1}, T^i_{k-1})
22                 for each i in {1,2, ..., p}
23                 do WINNER (I^i, w^i, T^i, ρ̄_a, β_a, myTemplates)
24                      if k = n − 1
```

25 **then if** $class(\mathbf{I}^i) = class(w^i)$

26 **then**

27 $myTemplates \leftarrow myTemplates \cup \{\mathbf{I}^i \wedge w^i\}$

28 **else** $newTemplate \leftarrow \mathbf{I}^i$

29 $index(newTemplate) \leftarrow newNodes + nodes$

30 $myTemplates \leftarrow myTemplates \cup \{\mathbf{I}^i, w^i\}$

31 $newNodes \leftarrow newNodes + 1$

32 **if** $newNodes > 0$

33 **then**

34 $nodes \leftarrow nodes + newNodes$

35 $myShare \leftarrow \left\lceil \frac{nodes}{n} \right\rceil$

36 SEND-NEXT $(k, n, \{(\text{none}, \text{none}, 0)\})$

37 RECV-NEXT $\left(k, n, \{w_{k+1}^i : i = 1, \ldots, 2p\}, newNodes_{k+1}\right)$

38 $myTemplates \leftarrow myTemplates \cup \{w_{k+1}^i : i = 1, \ldots, 2p\}$

**4. EXPERIMENTS**

The database used for testing the performance of the parallel, no-match tracking FAM was the Forest Covertype database, provided by Blackard, and donated to the UCI Repository. The experiments were run on OPCODE, a 96 node Beowulf cluster, connected by a fast Ethernet network. The database consists of a total of 581,012 patterns, each one associated with 1 of 7 different forest tree cover-types. The number of attributes of each pattern is 54, but this number is misleading since attributes 11 through 14 are actually a binary tabulation of the attribute *Wilderness-Area,* and attributes 15 to 54 ( 40 of them) are a binary tabulation of the attribute *Soil-Type*. The original database values are not normalized to fit in the unit hypercube (FAM requires normalization of input values, so that they lie in the interval [0, 1]). Hence, we normalized the input values. Patterns 1 through 512,000 were used for the training of the NMT-FAM. Patterns 561,001 to 581, 000 (20,000 of them) were used for testing. Training set sizes of $1000\ 2^i,\ i \in \{5, 6, \ldots, 9\}$ were used for the training of the no-match tracking FAM. The number of processors in the pipeline varied from *p=1* to *p*=32, in powers of 2. The metrics used to measure the performance of the pipelined approach were: (a) Classification performance of the pipelined no-match tracking FAM, and (b) Speed-up of the pipelined no-match tracking FAM versus sequential no-match tracking FAM. Results of the speed-up for this database can be seen in Figure 2. We observe from this figure that the speed-up is linear. For large training set sizes (i.e., 128,000 patterns) the speed-up is slightly above linear which suggests that the memory issues are a concern when few processes are in the pipeline. The classification performance of the Forest Covertype ranges from 70% to 79% as the training set size changes from 32,000 patterns to 512,000 patterns. It is worth mentioning that the classification performance of the no-match tracking FAM is comparable to the performance of the original FAM algorithm and better than the performance of other algorithms reported in the literature (best performance found was around 75%).

## 5. CONCLUSIONS

We have implemented a pipelined Fuzzy ARTMAP variant (called no-match tracking FAM). This FAM variant allowed us to focus on the parallelization of the competition process in Fuzzy ARTMAP. We have showed that this parallel implementation of the FAM variant is theoretically sound (results were omitted due to lack of space) and exhibits good workload balancing properties. We also showed experimentally (by working with the Covertype database) that this algorithm exhibited linear speed-up when the number of processors in the pipeline is increased. Furthermore, the generalization performance of this parallel no-matchtracking FAM was better than the performance of any other algorithm (from results reported in the literature) and similar to the performance of the original FAM. Finally, it is worth mentioning that, to the best of our knowledge, this is the first implementation of a Fuzzy ARTMAP-like algorithm on a Beowulf cluster.

**ACKNOWLEDGEMENTS:** Jose Castro and Michael Georgiopoulos would like to acknowledge the partial support of the NSF CRCD grant, no: 0203446. Georgios Anagnostopoulos and Michael Georgiopoulos would also like to acknowledge the partial support of the NSF CCLI grant, no: 0341601.

## REFERENCES

Anagnostopoulos, G. C., and Georgiopoulos, M., "Putting the utility of match-tracking in Fuzzy ARTMAP to the test," In Proceedings of the Seventh International Conference on Knowledge-based Intelligent Information Engineering, Vol. 2, pp. 1-6, KES, 2003.

Carpenter, G. A., Grossberg, S. Markuzon, N., Reynolds, J. H., Rosen, D. B., "Fuzzy ARTMAP: A neural network architecture for incremental learning of analog multi-dimensional maps," IEEE Transactions on Neural Networks, Vol. 3, No. 5. pp. 698-713, 1992.

Castro, J., "Modifications of the Fuzzy-ARTMAP Algorithms for Distributed Learning in Large Data Sets," PhD. Dissertation, University of Central Florida, Summer 2004.

Manolakos, E. S., "Parallel Implementation of ART1 neural networks on Processor Ring Architectures", in Parallel Architectures for Artificial Neural Networks, editors N. Sundararajan and P. Saratchandran, IEEE Computer Society Press, 1998.

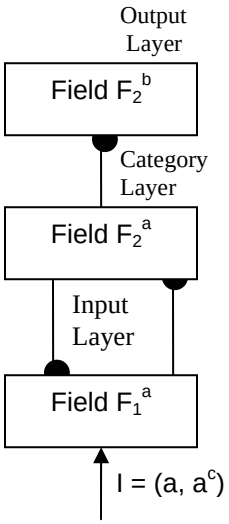


Figure 1. FAM Diagram.

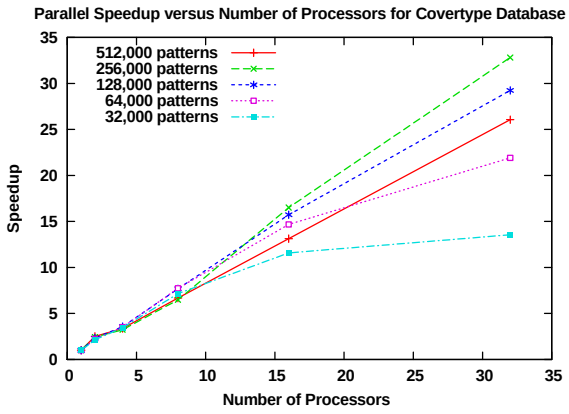


Figure 2. Performance of Pipelined NMT-FAM.