Pipelining of Fuzzy–ARTMAP (FAM) without Matchtracking (MT)

José 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

May 10, 2004

Abstract

Fuzzy ARTMAP (FAM) is a neural network architecture that can establish the correct mapping between real valued input patterns and their correct labels in a variety of classification problems. FAM has many desirable traits. Nevertheless, as the size of the data set grows to thousands, and hundreds of thousands data-points, 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 technique for the NMT-FAM, based on a pipeline, and show that a) the parallelized algorithm is equivalent to the sequential NMT-FAM, and b) the parallelization strategy achieves linear speedup in the order of p (number of processors). Experiments on the CoverType database support our results. 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 one of the fastest (in terms of training time) neural network algorithms, the Fuzzy–ARTMAP algorithm, tends to exhibit slow convergence time as the size of the network increases.

One way to address this problem is by the use of parallelization. Extensive research has been done on the properties of parallelization of feed–forward multi–layer perceptrons [11]. This is probably due to the popularity of this neural network architecture, and also because the backpropagation algorithm used to train these type of networks can be characterized mathematically by matrix and vector multiplications, mathematical structures that have been parallelized with extensive success.

For ART type neural networks we can find the work of Manolakos in [9] where he implemented the non-supervised learning ART1 architecture on a ring of processors. Another parallelization approach that has been used with ART and other types of neural networks is the systems integration approach where the neural network is not implemented on a network of computers but on parallel hardware. Zhang [13] shows how a fuzzy competitive neural network similar to ARTMAP can be implemented using a systolic array, and Asanović [3] et al., use a special purpose parallel vector processor SPERT-II to implement back-propagation and Kohonen neural networks. Nevertheless, no distributed implementation of ARTMAP or Fuzzy–ARTMAP was found in the literature.

Furthermore, none of the previous implementations address the issue of learning with very large data sets. Obviously, data mining algorithms have been introduced into the literature that address the large datasets issue with success. When the size of the databases used by these algorithms are in the millions of records it is of primary concern to bring down the complexity of the algorithm to polynomial, or logarithmic time. One case in point is the rule extraction RIPPER algorithm by Cohen [6] that scales in the order of $O(P(\log P)^2)$ ([7]), where P is the size of the training set.

Returning back to our algorithm of interest in this paper, it is also worth mentioning that Fuzzy ARTMAP is an online algorithm, capable of absorbing new information without disrupting previously learned knowledge and it also capable of learning from an infinite stream of immediately available data. Fuzzy–ARTMAP has many desirable characteristics, but would require modifications for the handling of large data sets. Fuzzy–ARTMAP has already demonstrated it's potential for moderately sized databases [2], but Fuzzy–ARTMAP s performance deteriorates when the training set grows. Furthermore, knowing that Fuzzy–ARTMAP is already an online algorithm (a very desirable characteristic), it is of interest to pursue the study of Fuzzy–ARTMAP 's characteristics and its parallel variants in large data sets, while preserving its online properties.

This paper is organized as follows: Section 2 presents briefly the Fuzzy–ARTMAP architecture and Fuzzy–ARTMAP algorithm. In this section, we simplify the Fuzzy–ARTMAP algorithm and provide a pseudocode that will be the starting point of the parallelization. Section 3 presents the Anagnostopoulos' no–matchtracking Fuzzy–ARTMAP variant algorithm ([1]), which is used as the basis of our parallel pipelined algorithm. Section 4 discusses briefly the parallel platform on which the no-matchtracking Fuzzy–ARTMAP is implemented (Beowulf cluster of computers). Section 5 outlines the code of the parallel no–matchtracking Fuzzy–ARTMAP algorithm. Section 6 discusses some important properties of the parallel no-match tracking FAM implementation. Section 7 proceeds with experiments and results comparing performance and speedup of the parallel no–matchtracking Fuzzy–ARTMAP variant on the CoverType database. We finalize the article with a conclusions section (see section 8), where we summarize our results.

2 The Fuzzy–ARTMAP Architecture and Algorithm

The Fuzzy–ARTMAP neural network ([5]) belongs to a family of ART neural network architectures and has been proven to be one of the premier neural network architectures for classification problems. Some of the advantages that Fuzzy–ARTMAP has, compared to other neural network classifiers, are that it learns the required task fast, it has the capability to do on-line learning, and its learning structure allows the explanation of the answers that the neural network produces.

There have been many contributions to the ART literature over the last decade. We only refer to a limited number of them: ARTEMAP, Gaussian ARTMAP, dART, dARTMAP, ARTMAP-IC, Boosted ARTMAP, Micro-ARTMAP, Ellipsoid-ART/ARTMAP, and semi-supervised ART architectures. The above contributions revolve around modifications and enhancements of the original Fuzzy–ARTMAP architectures. However there are other, independent developments of similar ART-like structures, like Fuzzy Min-Max, LAPART2, and σ-FLNMAP. In this paper, our focus is to improve the speed of convergence of ART-like structures through a training network partitioning approach. We chose to demonstrate the effectiveness of our proposed in the Fuzzy–ARTMAP architecture, since all of the aforementioned variants have enough in common with Fuzzy–ARTMAP as to make the parallelization schemes we present in this paper applicable to them, as well.

A block diagram of the Fuzzy–ARTMAP algorithm can be seen in figure 1. The Fuzzy–ARTMAP algorithm presented here is concordant with the Simplified Fuzzy–ARTMAP (SFAM) found in [8], this is a simplified version of the Fuzzy–ARTMAP algorithm that retains all the Fuzzy–ARTMAP functionality for classification tasks.

Most of the work in SFAM is done by the templates layer. In this architecture, input patterns are presented in complement coded format ($\mathbf{I} = (a, a^c);\ a_i^c = 1 - a_i^c$) to the input layer (layer F_1^a of the Fuzzy–ARTMAP block diagram). Once an input pattern is presented the nodes in the template layer (layer F_2^a in the Fuzzy–ARTMAP block diagram), compete for the representation of this input pattern in a winner–take–all competition. Note that every node in layer F_2^a of Fuzzy–ARTMAP is represented by a vector of weights ($\mathbf{w}_j^a$), referred to as templates. Every template in layer F_2^a is associated with a unique category or label, except one; this one template is the template associated with what is called "an uncommitted node", because it has not coded any input patterns yet. This association of templates to labels happens through the interconnection weights of layer F_2^a (template layer) and layer F_2^b (referred to as output layer). It is worth pointing out that learning in the Fuzzy–ARTMAP architecture is coded into the weight vectors $\mathbf{w}_j^a$; these vectors represent, in a compressed way, the information pertaining to all the input patterns that chose and were coded by template $\mathbf{w}_j^a$ in the training phase of Fuzzy–ARTMAP .

The Fuzzy–ARTMAP network can operate in two distinct modes: *learning* mode and *performance* mode.

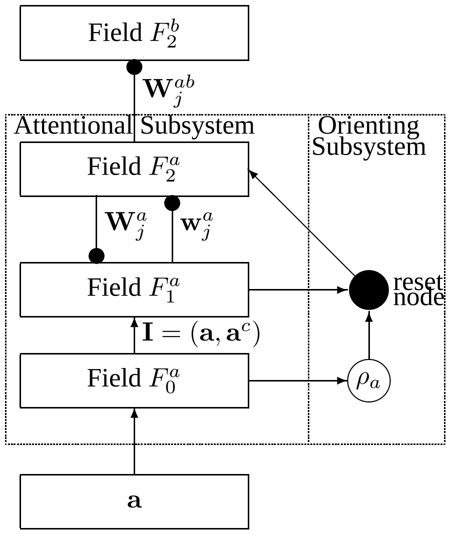

Figure 1: Block Diagram of the Fuzzy ARTMAP Architecture.

When it is operating in the learning mode the network is presented with an input/output pair $(\mathbf{I}, \mathbf{O})$ where $\mathbf{I}$ is a vector in $[0, 1]^{2M_a}$. The purpose of the network is then to learn the correct association of $\mathbf{I}$ to $\mathbf{O}$. When the network operates in the performance mode it is presented with an input $\mathbf{I}$ (not presented to it before) and it is required to predict the output $\mathbf{O}$. Usually, a training set is presented to the network (during its learning mode phase), and after the network is trained, its performance is evaluated on a different set, called test set (performance node phase). One of the advantages of Fuzzy–ARTMAP is that separating the learning and performance phases is not a requirement, and these phases can be mixed together, where the network either learns or performs, as needed.

In the Fuzzy–ARTMAP performance phase, the label (category) associated with the template that wins competition will be the one reported by the network as the associated category. If the uncommitted node is the winner then the network reports `no-classification`. In the learning phase, this process is not as simple, and three scenarios are possible:

1. the winning template is a committed template and of the same category as the input pattern.
2. the winning template is the uncommitted node.
3. the winning template is a committed node of the wrong category.

In the first case, the winning template learns the information that the new input pattern conveys. In the second case, the template of the uncommitted node learns the information that the new input pattern conveys; also this template is associated with the same label as the label of the new input pattern. Furthermore, a new uncommitted node is added to the Fuzzy–ARTMAP network to take care of similar type of situations in future input pattern presentations. In the third case we enter a process called matchtracking which resets the activation of the winning node and searches for another node to represent the input pattern; this search process continues until either a node of the correct label is found or the winning node is the uncommitted node.

There are three major operations that take place during the presentation of a training input/output pair (e.g., $(\mathbf{I}, \mathbf{O})$) to Fuzzy–ARTMAP . These are: (a) *Calculation of the activation values of the nodes in the template layer of Fuzzy–ARTMAP* . These values are denoted by $T(\mathbf{I}, \mathbf{w}_j^a, \beta_a)$, and they are fully defined in the next subsection (Fuzzy–ARTMAP pseudo-code). The template layer nodes are activated in Fuzzy–ARTMAP in the order of ascending activation values. (b) *Calculation of the vigilance ratio values of the nodes in the template layer of Fuzzy–ARTMAP* . These values are denoted by $\rho(\mathbf{I}, \mathbf{w}_j^a)$, and they are completely defined in the next subsection (Fuzzy–ARTMAP pseudo-code). Every activated, through operation (a), node in Fuzzy–ARTMAP needs to satisfy the vigilance criterion (i.e., its vigilance ratio needs to exceed a certain threshold value). (c) *The match-tracking mechanism, and change of the weights in Fuzzy–ARTMAP* . This match-tracking operation deactivates nodes in the template layer,initially chosen through operations (a) and (b), whose label is different than the label of the input pattern presented. The weights corresponding to an F_2^a node in Fuzzy–ARTMAP will change if the node has the maximum activation value, passes the vigilance test, and is of the correct label.

In all of the aforementioned Fuzzy–ARTMAP operations there is a specific operand involved, the *fuzzy min operand*, designated by the symbol $\wedge$. Actually, the fuzzy min operation of two vectors x, and y, designated as $x \wedge y$, 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 $|\cdot|$. In particular, $|x|$ is the size of a vector x and is defined to be the sum of its components.

```
FAM-ON-LINE-LEARNING($\left\{\mathbf{I}^1, \mathbf{I}^2, \ldots, \mathbf{I}^P\right\}, \bar{\rho}_a, \beta_a, \varepsilon_a$)
 1  $\mathbf{w}_0 \leftarrow \underbrace{(1, 1, \ldots, 1)}_{2M_a}$
 2  templates $\leftarrow \{\mathbf{w}_0\}$
 3  for each $\mathbf{I}^r$ in $\left\{\mathbf{I}^1, \mathbf{I}^2, \ldots, \mathbf{I}^P\right\}$
 4  do $\rho \leftarrow \bar{\rho}_a$
 5     repeat
 6             $T_{max} \leftarrow 0$
 7             status $\leftarrow$ NoneFound
 8             for each $\mathbf{w}_j^a$ in templates
 9             do if $\left[\rho(\mathbf{I}^r, \mathbf{w}_j^a) \geq \rho\right]$ and $\left[T(\mathbf{I}^r, \mathbf{w}_j^a, \beta_a) > T_{max}\right]$
10                   then
11                        $T_{max} \leftarrow T(\mathbf{I}^r, \mathbf{w}_j^a, \beta_a)$
12                        $j_{max} \leftarrow j$
13             if $\mathbf{w}_{j_{max}}^a \neq$ uncommitted
14               then if class($\mathbf{I}^r$) $=$ class($\mathbf{w}_{j_{max}}^a$)
15                       then status $\leftarrow$ Alloc
16                       else  status $\leftarrow$ Matchtracking
17                             $\rho \leftarrow \rho(\mathbf{I}^r, \mathbf{w}_{j_{max}}^a) + \varepsilon$
18     until status $\neq$ Matchtracking
19     if status $=$ Alloc
20       then
21             $\mathbf{w}_{j_{max}}^a \leftarrow \mathbf{w}_{j_{max}}^a \wedge \mathbf{I}$
22       else
23             templates $\leftarrow$ templates $\cup \{\mathbf{I}^r\}$
24  return templates
```

Figure 2: Fuzzy–ARTMAP online algorithm

2.1 The Fuzzy–ARTMAP Algorithm

This section contains an algorithmic description of Fuzzy–ARTMAP (see Figure 2 for the complete pseudocode). We only present the online version of the FAM algorithm. The pipeline approach that we propose is valid both for online and offline processing. The Fuzzy–ARTMAP algorithm needs the following parameters:

- *Patterns*: This is the set of learning patterns used to train the algorithm, each $\mathbf{I}^r \in$ *Patterns* is of the form $\mathbf{I}^r = (\mathbf{a}, \mathbf{a}^c)$ where $\mathbf{a} \in [0,1]^{M_a}$ and M_a represents the dimensionality of $\mathbf{a}$.
- $\bar{\rho}_a$: The baseline vigilance, a learning parameter that controls the size of the category templates, $\bar{\rho}_a \in [0,1]$.
- β_a: Also a learning parameter. The only restriction is that $\beta_a > 0$, but it is common to set it to a small value close to 0.

- ε: As it's notation reveals this is a very small number $\varepsilon > 0$. It is used to guarantee that ineligible templates are excluded from competition during the matchtracking process.

The Fuzzy–ARTMAP algorithm (learning phase) is shown in figure 2. The process we call matchtracking is being performed in the loop of lines 8 through 22. Once a template has won competition, we check if it is of the correct category (line 18) and if not, we increase the vigilance parameter ρ (line 21) so that the winning node is no longer eligible. Also notice that we repeat the learning process (lines 4 to 30) until we reach a maximum number of iterations or we perform a complete presentation of all the input patterns and no modifications are made to the network.

There a number of quantities that appear in the learning phase of Fuzzy–ARTMAP , as depicted in Figure 2. These quantities are defined below.

- $\mathbf{w}_j^a$: Weight vector in the Fuzzy–ARTMAP neural network that emanates from the template layer and converges to the input layer.
- $T(\mathbf{I}^r, \mathbf{w}_j^a, \beta_a)$: Activation of node with template $\mathbf{w}_j^a$ due to the presentation of input pattern $\mathbf{I}^r$.

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

- $\rho(\mathbf{I}^r, \mathbf{w}_j^a)$: Vigilance ratio, represents the level of match between an input patter $\mathbf{I}^r$ and the template $\mathbf{w}_j^a$

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

3 "No Matchtracking" Fuzzy–ARTMAP

Another modification that can be applied to the Fuzzy–ARTMAP algorithm is the elimination of the matchtracking process. This modification was originally proposed by Anagnostopoulos [1] and it was shown there that it can actually improve the classification performance of Fuzzy–ARTMAP on some databases, at the expense of creating more templates in F_2^a than the original Fuzzy–ARTMAP . Our interest in using this Fuzzy–ARTMAP variant lies in that it simplifies the Fuzzy–ARTMAP algorithm and allows us to concentrate on the parallelization of the competition loop of Fuzzy–ARTMAP . This no-match tracking Fuzzy–ARTMAP algorithm is depicted in figure 3.

```
FAM-NO-MATCHTRACKING-LEARNING($\left\{\mathbf{I}^1, \ldots, \mathbf{I}^P\right\}, \bar{\rho}_a, \beta_a$)
 1  $\mathbf{w}_0 \leftarrow \underbrace{(1, 1, \ldots, 1)}_{2M_a}$
 2  templates $\leftarrow \{\mathbf{w}_0\}$
 3  for each $\mathbf{I}^r$ in $\left\{\mathbf{I}^1, \mathbf{I}^2, \ldots, \mathbf{I}^P\right\}$
 4  do $T_{max} \leftarrow 0$
 5     $\mathbf{w}_{max} \leftarrow$ none
 6     for each $\mathbf{w}_j^a$ in templates
 7     do if $\left[\rho(\mathbf{I}^r, \mathbf{w}_j^a) \geq \bar{\rho}_a\right]$ and $\left[T(\mathbf{I}^r, \mathbf{w}_j^a, \beta_a) > T_{max}\right]$
 8           then
 9                $T_{max} \leftarrow T(\mathbf{I}^r, \mathbf{w}_j^a, \beta_a)$
10                $\mathbf{w}_{max}^a \leftarrow \mathbf{w}_j^a$
11     if $\mathbf{w}_{max}^a \neq \mathbf{w}_0$ and class$(\mathbf{I}^r)$ = class$(\mathbf{w}_{max}^a)$
12        then $\mathbf{w}_{max} \leftarrow \mathbf{w}_{max} \wedge \mathbf{I}^r$
13        else templates $\leftarrow$ templates $\cup \{\mathbf{I}^r\}$
14  return templates
```

Figure 3: Anagnostopoulos No–matchtracking Fuzzy–ARTMAP

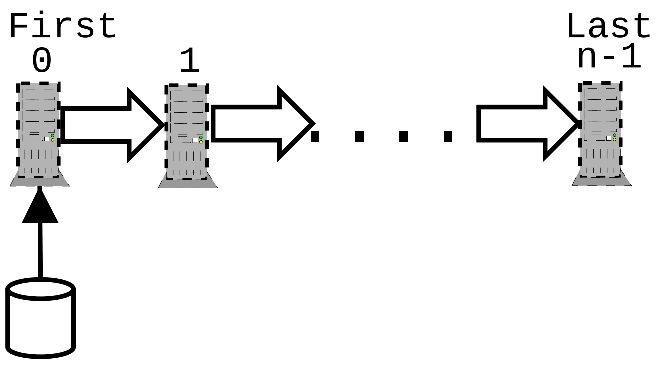

Figure 4: Pipeline Structure.

4 The Beowulf parallel platform

The Beowulf cluster of workstations is a network of computers where processes exchange information through the network's communications hardware. In general, the Beowulf cluster configuration is a parallel platform that has a high latency. Parallelization techniques in this platform are radically different from shared memory or vector machines. Our design is based on fixed packet size communication through the network. No network bandwidth would be gained by using variable sized packets since packets are more efficient when they are large and to find out the size of a packet a receiving process would have to incur an extra (and expensive) communication.

To find the optimum packet size for our experiments, we used a large database of patterns, of dimensionality 55. We transmitted information in the pipeline using different packet sizes. The lowest transmission time, regardless of the number of processors in the pipeline, occurred when the packet size was in between 64 to 128 patterns. Equating this number to memory size we get an optimum packet size for the cluster in the vicinity of:

$$(64 \ldots 128) \times 55 \times 4 = (14080 \ldots 28160)\texttt{Bytes} \tag{3}$$

5 Parallel, no matchtracking, Fuzzy–ARTMAP implementation

Anagnostopoulos' Fuzzy–ARTMAP variant is particularly amenable to a production–line style pipeline parallel implementation since patterns can be evenly distributed among the nodes in the pipeline. A depiction of the pipeline is shown in figure 4. The elimination of matchtracking makes the learning of a 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 we will introduce the following definitions:

- 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 current packet in the pipeline. $i \in \{1, 2, \ldots, p\}$.
- $\mathbf{w}^i$: 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 (it's own plus the other processors).
- *myShare*: amount of templates that the current process should have.
- $\mathbf{w}^i_{k-1}$: template i coming from previous process in the pipeline.
- $\mathbf{w}^i_{k+1}$: template i coming from next process in the ring.
- $\mathbf{w}^i$: template i going to next process in the ring.

- $\mathbf{w}^i_{to(k-1)}$: template i going to previous process in the pipeline.
- $\mathbf{I}.class$: class label associated with a given input pattern.
- $\mathbf{w}.class$: class label associated with a given template.
- *index*$(\mathbf{w})$: sequential index assigned to a 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.

The algorithm itself is shown in figure 5. The initialization procedure which initializes the templates to nulls is not shown.

6 Properties of Parallel, no matchtracking, Fuzzy–ARTMAP

To guarantee the correctness of the Parallel, no matchtracking Fuzzy–ARTMAP algorithm we developed a series of fourteen properties distinguished in performance and correctness properties. We only discuss two of these important properties, and we are presenting them in the form of theorems. For instance, Theorem 6.1 states that only one copy of the template will exist in the distributed system guaranteeing that no update conflicts or *stale* templates exist in the system. Furthermore, Theorem 6.2 defines a useful upper bound on the variance in the number of templates that each process possesses. Notice that this bound is independent of the pipeline depth, guaranteeing that if we add more processes to the pipeline we will get a better performance.

Theorem 6.1 *Template uniqueness guarantee*
A template in the neural network will reside in one processor and one processor only.

Theorem 6.2 *Workload balance variance bound*
In a pipeline with an arbitrary number of processors and a downstream packet size p, the standard deviation of the number of templates that each processor owns cannot exceed

$$\frac{p}{2\sqrt{3}} \tag{4}$$

If, for example, we use a packet size of 64 patterns, then the worst possible standard deviation in the value of *myShare* would not exceed

$$\frac{64}{2\sqrt{3}} = \frac{32}{\sqrt{3}} = 18.4752$$

regardless of the pipeline size n.

7 Experiments

The database used for testing was the Forest CoverType database provided by Blackard [4], and donated to the UCI Machine Learning Repository [12]. 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 to 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. Thus, we transformed the data to achieve this. There are no omitted values in the data.

Patterns 1 through 512,000 were used for training. The test set consisted of patterns 561,001 to 581,000. Classification performance of different machine learning algorithms for this database have been reported in the range of 75%.

Training set sizes of $1000 \times 2^i, i \in \{5, 6, \ldots, 9\}$, that is 32,000 to 512,000 patterns were used for the training of Fuzzy–ARTMAP and pipelined no matchtracking FAM. The test set size, as mentioned above, was fixed at 20,000 patterns. The number of processors in the pipeline varied from $p = 1$ to $p = 32$, in powers of 2.

Table 1: Classification performance of no–matchtracking *Fuzzy*–ARTMAP

Number of Patterns	Classification %
32,000	70.29
64,000	74.62
128,000	75.06
256,000	77.29
512,000	79.28

To avoid additional computational complexities in the the experiments (beyond the one that the size of the training set brings along) the values of the ART network parameters $\bar{\rho}_a$, and β_a were fixed (i.e., the values chosen were ones that gave reasonable results). For every combination of $(p, P) =$ (pipeline size, training set size) values we conducted 12 independent experiments (training and performance phases), corresponding to different orders of pattern presentations within the training set. All results reported are averages over the 12 runs.

All the tests where conducted on the `OPCODE` Beowulf cluster of workstations [10] of the Institute for Simulation and Training. This cluster consists of 96 9Athlon MP 1500+ nodes, each with 512MB of RAM each. This configuration guaranteed identical conditions on all runs, parallel and sequential. Optimal interprocess transmission packet size used as calculated by the experiment in section 4 was 64 (template, input–pattern) pairs per transmission.

The metrics used to measure the performance of the pipelined approach were:

1. Classification performance of pipelined no matchtracking FAM.
2. Speedup of pipelined no matchtracking FAM versus sequential no match-tracking FAM.

Results for the speedup for this database can be seen in figure 7. We can see that the speedup is close to linear. For large training set sizes the speedup is slightly above linear (i.e. 512,000 patterns) which suggests that memory issues are a concern when few processes are in the pipeline.

Classification performance is shown on table 1. It is worth noting that the classification of the no–matchtracking FAM variant (shown in Table 1) is comparable to the performance of the original FAM algorithm (these results are not shown on the table), and better than the performance of other algorithms reported for this database.

8 Conclusions

We implemented a pipelined *Fuzzy*–ARTMAP variant, referred to as no-match tracking FAM. The no-match tracking FAM is one of the match-tracking FAM variants introduced by Anagnostopoulos ([1]). This version of Fuzzy–ARTMAP allowed us to concentrate on the parallelization of the competition loop in Fuzzy–ARTMAP . We demonstrated experimentally that this variants classification performance is comparable with the performance of *Fuzzy*–ARTMAP. We have also formally showed that the algorithm is well behaved and has good workload balancing properties. The algorithm exhibited linear speedup when the number of processors is increased. This makes our Fuzzy–ARTMAP pipelined implementation suitable for data-mining applications, where the size of the database is large. To the best of our knowledge, this is the first implementation of a *Fuzzy*–ARTMAP like classification algorithm on a BEOWULF cluster.

Acknowledgment

The authors would like to thank the Institute of Simulation and Training and the Link Foundation Fellowship program for partially funding this project. This work was also supported in part by the National Science Foundation under a CRCD grant (no. 0203446).Michael Georgiopoulos also acknowledges the support of the NSF CCLI grant 0341601).

References

[1] G. C. Anagnostopoulos, “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, University of Oxford, UK. KES’03, 2003, pp. 1–6.

[2] G. Anagnostopoulos, “Novel approaches in adaptive resonance theory for machine learning,” Ph.D. dissertation, Computer Engineering, UCF, 2000.

[3] K. Asanović, J. Beck, B. Kingsbury, N. Morgan, D. Johnson, and J. Wawrzynek, *Parallel Architectures for Artificial Neural Networks: Paradigms and Implementations*. IEEE Computer Society Press and John Wiley & Sons, 1998, ch. Training Neural Networks with SPERT-II.

[4] J. A. Blackard, “Comparison of neural networks and discriminant analysis in predicting forest cover types,” Ph.D. dissertation, Department of Forest Sciences, Colorado State University, 1999.

[5] G. A. Carpenter, S. Grossberg, and J. H. Reynolds, “Fuzzy ART: An adaptive resonance algorithm for rapid, stable classification of analog patterns,” in *International Joint Conference on Neural Networks, IJCNN’91*, vol. II, IEEE/INNS Inc. Seattle, Washington: IEEE–INNS–ENNS, 1991, pp. 411–416.

[6] W. W. Cohen, “Fast effective rule induction,” in *Proceedings of the Twelfth International Conference on Machine Learning*, M. Kaufmann, Ed., San Francisco, California, 1995.

[7] T. G. Dietterich, “Machine-learning research, four current directions,” *AI Magazine*, Winter 1997.

[8] T. Kasuba, “Simplified Fuzzy ARTMAP,” *AI Expert*, pp. 18–25, November 1993.

[9] E. S. Manolakos, *Parallel Architectures for Neural Networks: Paradigms and Implementations*. IEEE Computer Society Press and John Wiley & Sons, 1998, ch. Parallel Implementation of ART1 Neural Networks on Processor Ring Architectures.

[10] P. Micikevicius, “Scerola parallel cluster,” http://www.cs.ucf.edu/courses/cda5110/scerola/guide-scerola.html, 2003.

[11] J. Torresen and S. Tomita, *Parallel Architectures for Artificial Neural Networks: Paradigms and Implementations*. IEEE Computer Society Press and John Wiley & Sons, November 1998, ch. A Review of Parallel Implementations of Backpropagation Neural Networks, pp. 41–118.

[12] University of California, Irvine, “Uci machine learning repository,” http://www.icf.uci.edu/mlearn/MLRepository.html, 2003.

[13] D. Zhang, *Parallel VLSI Neural Systems Design*. Springer, 1998.

```
PROCESS$(k, n, \bar{\rho}_a, \beta_a, p)$
 1  INIT$(p)$
 2  while continue
 3  do
 4      while $|myTemplates| > myShare$
 5      do
 6          EXTRACT-TEMPLATE $\left(myTemplates, \left\{\mathbf{w}^i_{to(k-1)}\right\}\right)$
 7      SEND-NEXT $\left(k, n, \left\{\left(\mathbf{w}^i, \mathbf{I}^i, T^i\right) : i = 1, \ldots, p\right\}\right)$
 8      RECV-NEXT $\left(k, n, \left\{\mathbf{w}^i_{k+1} : i = 1, \ldots, 2p\right\}, newNodes_{k+1}\right)$
 9      SEND-PREV $\left(k, \left\{\mathbf{w}^i_{to(k-1)} : i = 1, \ldots, 2p\right\}, newNodes\right)$
10      RECV-PREV $\left(k, \left\{\left(\mathbf{w}^i_{k-1}, \mathbf{I}^i_{k-1}, T^i_{k-1}\right) : i = 1, \ldots, p\right\}\right)$
11      $newNodes \leftarrow newNodes_{k+1}$
12      $\mathcal{S} \leftarrow \left\{\mathbf{w}^i_{k+1}\right\}$
13      for each $i$ in $\{1, 2, \ldots, p\}$
14      do WINNER$(\mathbf{I}^i, \mathbf{w}^i, T^i, \bar{\rho}_a, \beta_a, \mathcal{S})$
15      $myTemplates \leftarrow myTemplates \cup \mathcal{S}$
16      if $\mathbf{I}^i_{k-1} = \texttt{EOF}$
17         then $continue \leftarrow \texttt{FALSE}$
18         else $\mathcal{S} \leftarrow \left\{\mathbf{w}^i_{to(k-1)}\right\}$
19                for each $i$ in $\{1, 2, \ldots, p\}$
20                do WINNER$(\mathbf{I}^i_{k-1}, \mathbf{w}^i_{k-1}, T^i_{k-1}, \rho_a, \beta_a, \mathcal{S})$
21                    $\left(\mathbf{I}^i, \mathbf{w}^i, T^i\right) \leftarrow \left(\mathbf{I}^i_{k-1}, \mathbf{w}^i_{k-1}, T^i_{k-1}\right)$
22                for each $i$ in $\{1, 2, \ldots, p\}$
23                do WINNER$(\mathbf{I}^i, \mathbf{w}^i, T^i, \bar{\rho}_a, \beta_a, myTemplates)$
24                    if $k = n - 1$
25                       then if $class(\mathbf{I}^i) = class(\mathbf{w}^i)$
26                               then
27                                    $myTemplates \leftarrow myTemplates \cup \{\mathbf{I}^i \wedge \mathbf{w}^i\}$
28                               else $newTemplate \leftarrow \mathbf{I}^i$
29                                    $index(newTemplate) \leftarrow newNodes + nodes$
30                                    $myTemplates \leftarrow myTemplates \cup \{\mathbf{I}^i, \mathbf{w}^i\}$
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 $(k, n, \{(\texttt{none}, \texttt{none}, 0)\})$
37  RECV-NEXT $\left(k, n, \left\{\mathbf{w}^i_{k+1} : i = 1, \ldots, 2p\right\}, newNode_{k+1}\right)$
38  $myTemplates \leftarrow myTemplates \cup \left\{\mathbf{w}^i_{k+1} : i = 1, \ldots, 2p\right\}$
```

Figure 5: Pipelined no-match tracking Fuzzy–ARTMAP implementation for parallel processing

```
WINNER($\mathbf{I}, \mathbf{w}, T, \rho_a, \beta_a, \mathcal{S} = \{\mathbf{w}^i\}$)
 1  $idx \leftarrow -1$
 2  for each $\mathbf{w}^i$ in $\mathcal{S}$
 3  do if $[\rho(\mathbf{I}, \mathbf{w}^i) \geq \rho_a]$
 4       then
 5           if $[T(\mathbf{I}, \mathbf{w}^i, \beta_a) > T]$
 6             then
 7                 $T \leftarrow T(\mathbf{I}, \mathbf{w}^i, \beta_a)$
 8                 $idx \leftarrow i$
 9             else  if $[T(\mathbf{I}, \mathbf{w}^i, \beta_a) = T]$ and $index(\mathbf{w}^i) < index(\mathbf{w})$
10                     then $T \leftarrow T(\mathbf{I}, \mathbf{w}^i, \beta_a)$
11                          $idx \leftarrow i$
12  if $idx \neq -1$
13     then
14         EXTRACT($\mathbf{w}^{idx}, \mathcal{S}$)
15         ADD($\mathbf{w}, \mathcal{S}$)
16         $\mathbf{w} \leftarrow \mathbf{w}^{idx}$
17         return TRUE
18     else
19         return FALSE
```

Figure 6: Utility function to find best candidate template in a template list. Needed by parallel no-match tracking Fuzzy–ARTMAP pipelined implementation

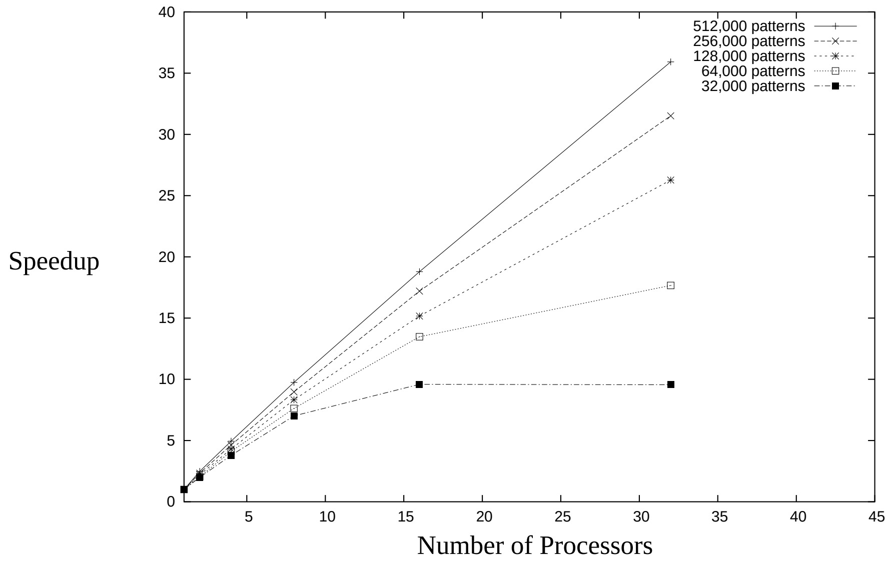

Figure 7: Speedup of Forest CoverType database for different training set and pipeline sizes