General Mathematics Vol. 12, No. 3 (2004), 37–52

An efficient ART1 Algorithm based on vector computation

Nicolae Popoviciu

Abstract

The work describes the ART1 (Adaptive Resonance Theory) algorithms in a vector form, not element by element as usually is done and gives all the dimensions for the vectors one used. This simplify the algorithm description and its using.

We present in a compacted vector form two approaches of ART1: version 1 based on [3] and version 2 based on [2] and we obtain two ART1 algorithms. Then we apply both algorithms at the same numerical example. Version 2 supplies a better result.

2000 Mathematical Subject Classification: 68T05, 82C323.

Keywords: ART1, resonance, vigilance test.

1 Notations and General Framework

1.1 Notations

We use the general notations of Artificial Neural networks (ANN) with small special meanings. The set of input vectors (input data, learning data, input patterns) is

$$I(x) = \{x(1), x(2), ..., x(N)\},\ x(t) \in \mathbb{R}^n,\ t = 1, N;\ t(\text{time}) \text{ natural number.}$$

We note that the sort writing $t = 1, N$ is equivalent with $t = 1, 2, ..., N$ and the writing $1 \leq t \leq N$ means one or more fixed values of t from 1 to M. The same rule is applied for any other numbering index.

In Adaptive Resonance Theory 1 (ART1) each input vector has binary components.

$F1$ is the input (comparison) layer with n neurons $F1_1, ..., F1_i, ..., F1_n$.

$F2$ is the output (competitive) layer with M neurons $F2_1, ..., F2_j, ..., F_M$.

The weights from neuron i of $F1$ layer towards the neuron j of $F2$ layer (feed forward or bottom-up) are

$$w_{i,j}(t), i = 1, N; j = 1, M; W(t) = (w_{i,j}(t)) \text{ of type } n \times M.$$

Each **column** in the matrix $W(t)$ is a column vector $w_j(t) \in \mathbb{R}^n$, $j = 1, M$.

$$w_j(t) = (w_{1j}(t)...w_{ij}(t)...w_{nj}(t))^T,\ \text{T - transpose.}$$

The weights from neuron j of F2 layer towards the neuron i of F1 layer (feed back or top-down) are

$$v_{j,i}(t), j = 1, M; i = 1, n; V(t) = (v_{ji}(t)) \text{ of type } M \times n.$$

Each **line** in the matrix $V(t)$ is a column vector $v_j(t) \in \mathbb{R}^n$, $j = 1, M$.

$$v_j(t) = (v_{j1}(t)...v_{ji}(t)...v_{jn}(t))^T.$$

Hence the vectors $w_j(t)$ and $v_j(t)$ have the same dimension n.

We denote by $o(t) \in \mathbb{R}^M$ a (fire) vector having one component for each $F2_j$ neuron. If the neuron j is “on” (enabled), then $o_j(t) = 1$ and if the neuron j is “off” (disabled), then $o_j(t) = 0$. The vector $\theta \in \mathbb{R}^M$ is the null vector of the space.

For any two vectors u and v belonging to the same vector space, let us say $u, v \in \mathbb{R}^n$, we use the usual notations

$< u, v >= u \cdot v = u^T v \in \mathbb{R}$, scalar product;

$u * v = (u_1 v_1 ... u_i v_i ... u_n v_n)^T \in \mathbb{R}^n$, piecewise product (component by component)

$$u \wedge v \in \mathbb{R}^n \quad \text{(component - wise minimum)} \tag{1}$$

This means the minimum on each pair of components $\min\{u_i; v_i\}, i = 1, n$.

We use 1-norm of vector u defined by $||u||_1 = ||u|| = \sum_{i=1}^{n} |u_i|$.

The real values $\rho \in (0, 1)$ and $\alpha \in (0, 1)$ are vigilance parameter and learning rate, respectively.

1.2 ART 1 Algorithms

The ART1 is a unsupervised and competitive learning. It performs data clustering on the input data $I(x)$. We denote the classes by $A1, ..., AM$. hence we have a map $f : \mathbb{R}^n \to \mathbb{R}^M$. By ART1 algorithms, learned vectors (input patterns) are retained even while new input vectors are learned. This means the plasticity-stability dilemma is solved.

There are more approaches of ART1 having small or rather big differences between them. We present two approaches of ART1, put in order the steps of two algorithms and apply them at the same numerical example. Then some conclusions are formulated.

For both algorithms **the initialization of parameters** is: $n, M, N; \rho, \alpha;$

$$w^0_{i,j} = w_{i,j}(1) = \frac{1}{1+n}, v^0_{j,i} = v_{j,i}(1) = 1 \text{ for } i = 1, n; j = 1, M.$$

2 ART1 Lippmann's Algorithm in the Krose-Smagt's Version

The Lippmann's algorithm (1987) is described by Krose and Smagt (1996) in [3]. In this section we write the ART1 algorithm in the **vector form** and we obtain the following steps.

Step 1. Do the initialization of parameters as it was described above (except α). At the beginning of the work, the fire vector $o(1)$ has all component equal 1 i.e. $o_j(1) = 1$.

Step 2. Execute a **DO** cycle for $t = 1, N$

$L1$ DO $t = 1, N$

Read an input vector $x(t)$.

$L2$ Compute the activation values for all the enabled neurons j of $F2$:

$$net_j(t) =< w_j(t), x(t) >, \text{ only if } o_j(t) = 1.$$

Select the winning neuron $k, 1 \leq k \leq M$ i.e. $\max\{net_j(t) | 0_j(t) = 1\} = net_k(t)$.

Remark 1. *If the maximum is not unique and there is a tie for the winning neuron k in F2 layer, then choose a* **special rule** *to break the tie, let us say the winner k is the second between the equals .*

Process the vigilance test for $F2_k$ output neuron :

$$r = \frac{< v_k(t), x(t) >}{||x(t)||} \tag{2}$$

If $r > \rho$ (there exists resonance) then go to label L3; otherwise (it is not resonance) go to label L4.

L3 Update the vectors $v_k(t)$ and $w_k(t)$ as follows

$$v_k(t+1) = v_k(t) * x(t) \tag{3}$$

$$w_k(t+1) = \frac{v_k(t+1)}{0,5 + ||v_k(t+1)||} \tag{4}$$

$$v_j(t+1) = v_j(t), w_j(t+1) = w_j(t) \text{ for all } j = 1, M, j \neq k.$$

Put $0_j(t+1) = 1, j = 1, M$ and store the input pattern in the class Ak.

Go to label L6 (the end of DO cycle).

L4 Disable the output neuron $F2_k$ i.e. $o_k(t) = 0$ and go to label L2. When $o(t) = \theta$ the network rejects the input vector $x(t)$ and it is stored in a reject set $RI(x)$. Go to label L5.

L5 Put the old weights on the positions of new weights and enable all the output neurons, namely

$$v_j(t+1) = v_j(t), w_j(t+1) = w_j(t), o_j(t+1) = 1 \text{ for all } j = 1, M,$$

L6 CONTINUE go to label L1.

Step 3. We repeat the Step 2 with the input set $RI(x)$ for a new N (cardinal of reject set).

Step 4. Print the content of classes and the matrixes W and V.

We call this algorithm **ALGOART1Version1**.

The work [1] mentions a Georgiopoulos Theorem [4] which assure us that the ART1 algorithm converges.

3 ART1 Algorithm in the Jain's Version

Again we use the **vector form** to describe the ART1 algorithm and we mention all the vector dimensions. In this version [2] we use the operation (1) and other formula instead of (3) to update the weights for the winner $F2_k$. It appears a new vector $A(t) \in \mathbb{R}^n$ (see (5) below). Also we use the learning rate α.

Step 1. Is the same as in section II.

Step 2. Execute a DO cycle for $t = 1, N$.

L1 DO $t = 1, N$

Read an input vector $x(t)$.

L2 Compute the activation values for all the enabled neurons j of F2:

$$net_j(t) =< w_j(t), x(t) >, \text{ only if } o_j(t) = 1.$$

Select the winning neuron $k, 1 \leq k \leq M$ i.e. $= \max\{net_j(t) | o_j(t) = 1\} = net_k(t)$.

It remains the same remark as in section II.

Process the vigilance test for $F2_k$ output neuron:

$r = \dfrac{< v_k(t), x(t) >}{||x(t)||}$; if $r > \rho$ (there exists resonance) then go to label L3; otherwise (it is not resonance) go to label L4.

L3 Update the vectors $v_k(t)$ and $w_k(t)$ as follows

$$A(t) = x(t) \wedge V^T(t)o(t) \tag{5}$$

$$v_k(t+1) = v_k(t) + \alpha[A(t) - v_k(t)] \tag{6}$$

$$w_k(t+1) = \frac{v_k(t+1)}{0,5 + ||v_k(t+1)||}$$

$$v_j(t+1) = v_j(t), w_j(t+1) = w_j(t) \text{ for all } j = 1, M, j \neq k.$$

Put $o_j(t+1) = 1,\ j = 1, M$ and store the input pattern in the class Ak.

Go to label L6 (the end of DO cycle).

L4 Disable the output neuron $F2_k$ i.e. $o_k(t) = 0$ and go to label L2. When $o(t) = \theta$ the network rejects the input vector $x(t)$ and it is stored in a reject set $RI(x)$. Go to label L5.

L5 Put the old weights on the positions of new weights and re-enable all the output neurons, namely

$$v_j(t+1) = v_j(t), w_j(t+1) = w_j(t), o_j(t+1) = 1 \text{ for all } j = 1, M.$$

L6 CONTINUE go to label L1.

Step 3. We repeat the Step 2 with the input set $RI(x)$ for a new N (cardinal of reject set).

Step 4. Print the content of classes and the matrixes W and V.

We call this algorithm **ALGOART1Version2**.

4 One Numerical Example and Two Algorithms

We would like to classify in **even** or **odd** the numbers 1, 2, 3, 4, 5, 6, 7 given in the usual binary-coded-decimal format [5]. hence the input data are given by the set $I(x)$ having the form

$$I(x) = \{x(1), x(2), x(3), x(4), x(5), x(6), x(7)\},$$

$$x(1) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, x(2) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, x(3) = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, x(4) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},$$

$$x(5) = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, x(6) = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, x(7) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Hence $n = 3$ i.e. the input layer F1 contains 3 neurons; $N = 7$ and $t = 1, 7$. We call A1 the class containing the even numbers and A2 the class with odd numbers. Consequently, $M = 2$ and the output layer contains 2 neurons. The other initializing parameters are $\rho = 0,3$ (vigilance parameter), $\alpha = 0,9$ (learning rate) and the weights for time $t = 1$

$$W(1) = \begin{pmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \\ 1/4 & 1/4 \end{pmatrix}, V(1) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, o(1) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Our computation task is to find the weights matrices $W(8) = W^*$, $V(8) = V^*$, when the neural network is considered to be learned.

The numerical results for ALGOART1Version1.

The special rule: the winner is the second between the equals.

Epoch 1. We use **the natural order** of input vectors and $\rho = 0,3$.

$t = 1; x(1); net_1(1) =< w_1(1) >= 1/4; net_2(1) =< w_2(1), x(1) >= 1/4;$

$\max\{1/4; 1/4\} = 1/4;$ (indecision, tie); take $k = 2$; vigilance test

$r = \frac{< v_2(1), x(1) >}{||x(1)||} = \frac{1}{1} = 1 > \rho = 0.3$ (there exists resonance);

$x(1)$ is accepted by $F2_2; x(1) \in A2$; true; update the vectors w_2, v_2

$$v_2(2) = v_2(1) * x(1) = (001)^T; w_2(2) = \frac{v_2(2)}{0,5 + ||v_2(2)||} = (002/3)^T$$

$$W(2) = \begin{pmatrix} 1/4 & 0 \\ 1/4 & 0 \\ 1/4 & 2/3 \end{pmatrix}, V(2) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, o(2) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

$t = 2; x(2); \max\{1/4; 0\} = 1/4;$ (decision); winner $k = 1$; vigilance test

$$r = \frac{< v_1(2); x(2) >}{||x(2)||} = \frac{1}{1} = 1 > \rho = 0,3 \text{ (there exists resonance);}$$

$x(2)$ is accepted by $F2_1; x(2) \in A1$; true; update the vectors w_1, v_1

$$v_1(3) = v_1(2) * x(2) = (0\ 1\ 0)^T; w_1(3) = \frac{v_1(3)}{0,5 + ||v_1(3)||} = (0\ 2/3\ 0)^T$$

$$W(3) = \begin{pmatrix} 0 & 0 \\ 2/3 & 0 \\ 0 & 2/3 \end{pmatrix}; V(3) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; o(3) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

$t = 3; x(3); \max\{2/3; 2/3\} = 2/3;$ (indecison, tie); take $k = 2$; vigilance test :

$$r = \frac{< v_2(3), x(3) >}{||x(3)||} = \frac{1}{2} = 0.5 > \rho = 0.3 \text{ (resonance); } x(3) \in A2; \text{true;}$$

$$v_2(4) = v_2(3) * x(30 = (0\ 0\ 1)^T; w_2(4) = \frac{v_2(4)}{0.5 + ||v_2(4)||} = (0\ 0\ 2/3)^T$$

$$W(4) = W(3); V(4) = V(3); o(4) = (11)^T.$$

$t = 4; x(4); \max\{0; 0\} = 0$; (indecision, tie); we analyze both cases:

take $k = 1$; vigilance test $r = 0 < \rho = 0.3$ (isn't resonance); put $o_1(4) = 0$

and $net_2(4) = 0; \max\{0\} = 0; k = 2; r = 0 < \rho = 0.3$; (isn't resonance);

$o_2(4) = 0; x(4)$ is rejected by F2 layer.

$$W(5) = W(4); V(5) = V(4); o(5) = (1\ 1)^T.$$

$t = 5; x(5)$; (decision); $k = 2; r = 0.5 > \rho = 0.3$; resonance; $x(5) \in A2$; true update vectors w_2, v_2 and we obtain

$$W(6) = W(5); V(6) = V(5); o(6) = (1\ 1)^T.$$

$t = 6; x(6)$; (decision) $k = 1; r = 0.5 > \rho = 0.3$; resonance; $x(6) \in A1$; true; update w_1, v_1 and we obtain

$$W(7) = W(6); V(7) = V(6); o(7) = (1\ 1)^T.$$

$t = 7; x(7)$; (indecision, tie); take $k = 2; r = 1/3 > \rho = 0.3$; resonance; $x(7) \in A2$; true; update w_2, v_2 and we obtain

$$W(8) = \begin{pmatrix} 0 & 0 \\ 2/3 & 0 \\ 0 & 2/3 \end{pmatrix} = W*; V(8) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = V * .Stop.$$

The content od classes is $A1 = \{x(2), x(4),\ \ A2 = \{x(1), x(3), x(5), x(7),$ $x(4)$ is rejected. The network is learned by the weights matrices W^* and V^*. We can continue in the same way with Epoch 2 and obtain the same result as in Epoch 1.

Remark 2. *During the Epoch 1 let us suppose we use the input vectors in the following* **mixed order** $I(x) = \{x(7), x(1), x(6), x(2), x(5), x(3), x(4)\}$. *We obtain the same result as before and* $x(4)$ *is also rejected. Nevertheless there exist several small differences between the intermediate results of the Epoch 1 (See Table 1 below).*

The numerical results for ALGOART1Version2.

The special rule: the winner is the second between the equals.

Epoch 1. We use **the natural order** of input vectors and $\rho = 0.3$, $\alpha = 0.9$.

$$W(1) = \begin{pmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \\ 1/4 & 1/4 \end{pmatrix}, V(1) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, o(1) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

$t = 1; x(1); \max\{net_1(1); net_2(1) = \max\{1/4; 1/4\} = 1/4$; (indecision); take winner $k = 2$; vigilance test $r = \frac{1}{1} = 1 > \rho = 0.3$; resonance; $x(1) \in A2$; update v_2, w_2 by (5) and (6): $v_2(2) = v_2(1) + 0.9[A(1) - v_2(1)]$

$$A(1) = x(1) \wedge V(1)^T o(1) = (001)^T,$$

$$v_2(2) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 0.9 \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0.1 \\ 1 \end{pmatrix},$$

$$w_2(2) = \frac{v_2(2)}{0.5 + ||v_2(2)||} = \frac{1}{1.7} \begin{pmatrix} 0.1 \\ 0.1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.06 \\ 0.06 \\ 0.59 \end{pmatrix};$$

$$W(2) = \begin{pmatrix} 0.25 & 0.06 \\ 0.25 & 0.06 \\ 0.25 & 0.59 \end{pmatrix}, V(2) = \begin{pmatrix} 1 & 1 & 1 \\ 0.1 & 0.1 & 1 \end{pmatrix}, o(2) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

$t = 2; x(2); \max\{net_1(2), net_2(2) = \max\{0.25; 0.06\} = 0.25$ (decision), $k = 1$; vigilance test $r = \frac{1}{1} = 1 > \rho = 0.3$; resonance; $x(2) \in A1$; update v_1, w_1 as $v_1(3) = v_1(2) + 0.9[A(2) - v_1(2)]$,

$$A(2) = x(2) \wedge V(2)^T o(2) = (0\ 1\ 0)^T, v_1(3) = (0.1\ 1\ 0.1)^T,$$

$$w_1(3) = \frac{v_1(3)}{0.5 + ||v_1(3)||} = (0.06\ 0.59\ 0.06)^T;$$

$$W(3) = \begin{pmatrix} 0.06 & 0.06 \\ 0.59 & 0.06 \\ 0.06 & 0.59 \end{pmatrix}, V(3) = \begin{pmatrix} 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 1 \end{pmatrix}, o(3) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Remark 3. *It would be better to work with more than two decimal figures.*

$t = 3; x(3); \max\{0.65; 0.65\} = 0.65$ (indecision); take winner $k = 2$; vigilance test $r = \frac{< v_2(3), x(3) >}{||x(3)||} = \frac{1.1}{2} = 0.55 > \rho = 0.3$; resonance; $x(3) \in A2$; update v_2, w_2;

$$v_2(4) = v_2(3) + 0.9[A(3) - v_2(3)]; A(3) = x(3) \wedge V(3)^T o(3) = (0\ 1\ 1)$$

$$v_2(4) = (0.1\ 0.91\ 1)^T; w_2(4) = \frac{v_2(4)}{2.51} = (0.04\ 0.36\ 0.40)^T;$$

$$W(4) = \begin{pmatrix} 0.06 & 0.04 \\ 0.59 & 0.36 \\ 0.06 & 0.40 \end{pmatrix}, v(4) = \begin{pmatrix} 0.1 & 1 & 0.1 \\ 0.1 & 0.91 & 1 \end{pmatrix}, o(4) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

$t = 4; x(4); \max\{0.06; 0.04 = 0.06$; (decision); $k = 1$; vigilance test $r = \frac{< v_1(4), x(4) >}{||x(4)||} = \frac{0.1}{1} = 0.1 < \rho = 0.3$; isn't resonance; $o_1(4) = 0$;

$\max\{net_2(4)|o_2(4)=1\} = \max\{0.04\} = 0.04$; $k=2$; $r = \dfrac{< v_2(4), x(4) >}{||x(4)||} = \dfrac{0.1}{1} = 0.1 < \rho = 0.3$; isn't resonance; $o_2(4)=0$; $x(4)$ is rejected by the network.

$$W(5) = W(4), V(5) = V(4), o(5) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

$t=5; x(5); \max\{0.12; 0.08\} = 0.12$; decision; winner $k=1$; vigilance test $r = \dfrac{0.2}{2} = 0.1 < \rho = 0.2$; isn't resonance; $o_1(5) = 0$;

$$\max\{net_2(5)|o_5(5)=1\} = \max\{0.08\} = 0.08;\ \text{winner } k=2;$$

$r = \dfrac{< v_2(5), x(5) >}{||x(5)||} = \dfrac{1.1}{2} = 0.55 > \rho = 0.3$; resonance; $x(5) \in A2$; update v_2, w_2 with $o(5) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$; $A(5) = x(5) \wedge V(5)^T o(5) = (0.1\ 0.1)^T$,

$$v_2(6) = (0.1\ 0.09\ 1)^T, w_2(6) = \frac{v_2(6)}{1.69} = (0.06\ 0.05\ 0.59)^T$$

$$W(6) = \begin{pmatrix} 0.06 & 0.06 \\ 0.59 & 0.05 \\ 0.06 & 0.59 \end{pmatrix}, V(6) = \begin{pmatrix} 0.1 & 1 & 0.1 \\ 0.1 & 0.09 & 1 \end{pmatrix}, o(6) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

$t=6; x(6); \max\{0.65; 0.11\} = 0.65$; decision; winner $k=1$; vigilance test

$$r = \frac{1.1}{2} = 0.55 > \rho = 0.3;\ \text{resonance; } x(6) \in A1;\ \text{update } v_1, w_1$$

$$A(6) = (0.2\ 1\ 0), v_1(7) = (0.19\ 1\ 0.01)^T, w_1(7) = \frac{v_1(7)}{1.7} = (0.11\ 0.59\ 0)^T;$$

$$W(7) = \begin{pmatrix} 0.11 & 0.06 \\ 0.59 & 0.05 \\ 0.00 & 0.59 \end{pmatrix}, v(7) = \begin{pmatrix} 0.19 & 1 & 0.01 \\ 0.10 & 0.09 & 1 \end{pmatrix}, o(7) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$t = 7; x(7); \max(0.70; 0.701) = 0.701$; decision; winner $k = 2$; vigilance test $r = 0.39 > \rho = 0.3$; resonance; $x(7) \in A2$; update v_2, w_2

$$A(7) = (0.29\ 1\ 1)^T, v_2(8) = (0.27\ 0.91\ 1)^T, w_2(8) = (0.1\ 0.34\ 0.37)^T$$

$$W(8) = \begin{pmatrix} 0.11 & 0.10 \\ 0.59 & 0.34 \\ 0.00 & 0.37 \end{pmatrix} = W*, V(8) = \begin{pmatrix} 0.19 & 1.00 & 0.01 \\ 0.27 & 0.91 & 1.00 \end{pmatrix} = V*. \text{ Stop.}$$

Epoch 2. We test only the vector $x(4)$ which was rejected.

$$t = 8; x(4); o(8) = \begin{pmatrix} 1 \\ 1 \end{pmatrix};$$

$$net_1(8) =< w_1(8), x(4) >= 0.11, net_2(8) =< w_2(8), x(4) >= 0.1,$$

$$\max\{0.11; 0.1\} = 0.11;\ \text{decision};\ k = 1;\ \text{vigilance test};$$

$$r = \frac{< v_1(8), x(4) >}{||x(4)||} = \frac{0.19}{1} = 0.19 < \rho = 0.3;\ \text{isn't resonance};\ o_1(8) = 0$$

$$k = 2; r = 0.27 < \rho = 0.3;\ \text{isn't resonance};\ x(4)\ \text{is rejected.}$$

Hence $A1 = \{x(2); x(6), A2 = \{x(1); x(3); x(5); x(7)$.

Now we summarize the numerical results we have obtained. By ALGO-ART1Version1 we have Table 1 (indecision=indec; decision=dec)

$I(x)$	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$x(6)$	$x(7)$
max{net}	*indec*	*dec*	*indec*	*indec*	*dec*	*dec*	*indec*
class	*A2*	*A1*	*A2*	*rejected*	*A2*	*A1*	*A2*

3 *decisions*; 4 *indecisons*;

$I(x)$	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$x(6)$	$x(7)$
max{net}	*indec*	*dec*	*dec*	*dec*	*dec*	*indec*	*indec*
class	*A2*	*A2*	*A1*	*A1*	*A2*	*A2*	*rejected*

4 *decisions*; 3 *indecisons*.

By ALGOART1Version2 we obtained Table 2.

$I(x)$	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$x(6)$	$x(7)$
$max\{net\}$	$indec$	dec	$indec$	dec	dec	dec	dec
$class$	$A2$	$A1$	$A2$	$rejected$	$A2$	$A1$	$A2$

5 *decisions*; 2 *indecisons*

5 Conclusions

ALGOART1Version2 is more refined and it assures better numerical results than ALGOART1Version1.

References

[1] Caudell T. P., Healy M. J., *Lateral Priming Adaptive Resonance Theory (LAPART)-2: Innovation in ART*; Chapter 6 in recent Advances in ANN. design and Applications, Edited by Lakhms Jain and Ana Maria Fanelli, The CRC Press, 2000

[2] Jain K. Anil, Mao Jianchang, Mohiuddin K. M., *Artificial Neural Networks: A Tutorial*, IEEE, MArch, 1996.

[3] Krose Ben, Van der Smagt P., *An Introduction to Neural Networks*, Chapter 6, Eight Edition, University of Amsterdam, 1996.

[4] Georgiopoulos M., Heileman G. L., Huang L., *Neural Networks. Properties of Learning Related to Pattern Diversity in ART1*, Vol. 4, pp. 751-757, 1991.

[5] Murgan T. A. & Co., *Structuri de Retele Neurale Artificiale. Simulari in MATLAB*, Editura Polithenică, Bucharest, 1995.

[6] Rao B. V., *C++, Neural Networks and Fuzzy logic*, Chapter 10, MT-Books, IDG Books Worldwide, Inc. 1995.

Hyperion University of Bucharest
Calea Călăraşilor 169
Bucharest, Romania