Neurocomputing 131 (2014) 132–142

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Efficient training for dendrite morphological neural networks

Humberto Sossa *, Elizabeth Guevara

CIC-IPN, Av. Juan de Dios Batiz, S/N, Col. Nva. Industrial Vallejo, Mexico D.F. 07738, Mexico

ARTICLE INFO

Article history:
Received 30 November 2012
Received in revised form
16 October 2013
Accepted 17 October 2013
Communicated by A. Belatreche
Available online 14 November 2013

Keywords:
Dendrite morphological neural network
Efficient training
Pattern recognition
Classification

ABSTRACT

This paper introduces an efficient training algorithm for a dendrite morphological neural network (DMNN). Given p classes of patterns, C^k, $k=1, 2, \ldots, p$, the algorithm selects the patterns of all the classes and opens a hyper-cube HC^n (with n dimensions) with a size such that all the class elements remain inside HC^n. The size of HC^n can be chosen such that the border elements remain in some of the faces of HC^n, or can be chosen for a bigger size. This last selection allows the trained DMNN to be a very efficient classification machine in the presence of noise at the moment of testing, as we will see later. In a second step, the algorithm divides the HC^n into 2^n smaller hyper-cubes and verifies if each hyper-cube encloses patterns for only one class. If this is the case, the learning process is stopped and the DMNN is designed. If at least one hyper-cube HC^n encloses patterns of more than one class, then HC^n is divided into 2^n smaller hyper-cubes. The verification process is iteratively repeated onto each smaller hyper-cube until the stopping criterion is satisfied. At this moment the DMNN is designed. The algorithm was tested for benchmark problems and compare its performance against some reported algorithms, showing its superiority.

1. Introduction

Pattern classification is a relevant problem in Artificial Intelligence. If a machine needs to efficiently interact with its environment, this problem should be correctly solved. The pattern classification problem can be stated as follows: given a pattern $X=(x_1, x_2, \ldots, x_n)^T$, or a distorted version $\tilde{X}$, find, somehow, its membership class $C^1, C^2, \ldots, C^p$. Hundreds of proposals have been described in the literature. A group of them (the distance based methods) make use of a distance between pattern X and the prototype $Z^k, k=1, 2, \ldots, p$, of each class C^k, and the pattern is assigned to the class C^k, for which $d(X, Z^k)$ is minimal. Another group (the decision based methods) uses a set of separation functions: f_1, f_2, …, among classes. One more group (the probability based methods) utilizes the a priori conditional probabilities: $p(C^k|X)$, and assigns X to the class for which $p(C^i|X) > p(C^j|X), i \neq j$. The methods belonging to the syntactical approach make use of structural pattern descriptions to determine the index class of X. In short, those methods based on the Artificial Neural Network (ANN) approach utilize the knowledge codified into the weights of the ANN to classify X.

The so called dendrite morphological neural networks (DMNN) belong to the ANN approach. DMNN were first described by Ritter and colleagues in [19,22]. DMNN emerge as an improvement of classical morphological neural networks (MNN), originally introduced by Davidson in [5] and then re-discussed by Ritter and Sussner in [21]. Morphological neural networks are closely related to Lattice Computing [9], which can be defined as the collection of Computational Intelligence tools and techniques that either make use of lattice operators *inf* and *sup* for the construction of the computational algorithms and replace the multiplication algebra operator of the real numbers by the addition operator. Algorithms and applications of Lattice Computing can be found in [12,8,7,31,29,30,13,24,4,3,25,34,6,15,17,27]. Morphological perceptrons with competitive learning, a variation of standard morphological perceptrons are discussed in [28]. Processing at dendrites level and not only at the cell body level allows neurons to power their processing capacities [26]. This fact is taken into account by Ritter and colleagues in the DMNN proposal.

A key issue in the design of a DMNN is its training; this is in the selection of the number of dendrites and the values of synaptic weights for each dendrite. Diverse algorithms to automatically train a DMNN can be found in [19,22,2,32,20,23,4].

In this paper, a novel algorithm for the automatic training of a DMNN is presented. At the first step, the algorithm takes the complete training set and opens a hyper-cube such that all the patterns rest inside. The size of HC^n can be chosen such that the border elements remain in some of the faces of HC^n, or can be chosen for a bigger size by adding a margin M on each side of the hyper-cube. An evolutive algorithm was used to determine the optimal value of M. At the second step, the algorithm divides the hyper-cube into 2^n smaller hyper-cubes, (with n the dimensionality) and verifies if each hyper-cube contains only patterns from

* Corresponding author. Tel.: +52 5518811983.
E-mail addresses: hsossa@cic.ipn.mx, humbertosossa@gmail.com (H. Sossa).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.10.031

one class C. If this is the case, the algorithm provides the designed DMNN and stops, otherwise it divides each smaller hyper-cube until the stopping criterion is satisfied. For testing, the hard limiter activation function of the original DMNN was changed by the maximum function to improve the results.

The rest of the paper is organized as follows. In Section 2 the basics and the most important concepts on dendrite morphological neural networks are given. Section 3 is dedicated to explain the philosophy and functionality of the proposed training algorithm. Section 4 is focused to present the experimental results where the proposed training algorithm is tested and compared with other reported methods by using some examples. Finally, Section 5 is oriented to provide the conclusions and directions for further research.

2. Basics on dendrite morphological neural networks

Morphological neural networks use lattice operations $\vee$ (maximum), or $\wedge$ (minimum), and $+$ from the semi-rings $(\mathbb{R}_{-\infty}, \vee, +)$ or $(\mathbb{R}_{\infty}, \wedge, +)$ where $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$ and $\mathbb{R}_{\infty} = \mathbb{R} \cup \{\infty\}$. The neuron computation in a MNN for input $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is given by

$$\tau_j(\mathbf{x}) = a_j \bigvee_{i=1}^{n} b_{ij}(x_i + w_{ij}) \tag{1}$$

or

$$\tau_j(\mathbf{x}) = a_j \bigwedge_{i=1}^{n} b_{ij}(x_i + w_{ij}) \tag{2}$$

where $b_{ij} = \pm 1$ denotes if the ith neuron causes excitation or inhibition on the jth neuron, $a_j = \pm 1$ denotes the output response (excitation or inhibition) of the jth neuron to the neurons whose axons contact the jth neuron and w_{ij} denotes the synaptic strength between the ith neuron and the jth neuron. Parameters b_{ij} and a_j take $+1$ or -1 value if the ith input neuron causes excitation or inhibition to the jth neuron.

The computation performed by the kth dendrite can be expressed by the formula

$$D_k(\mathbf{x}) = a_k \bigwedge_{i \in I} \bigwedge_{l \in L} (-1)^{1-l}(x_i + w_{ik}^l) \tag{3}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ corresponds to the input neurons, $I \subseteq \{1, \ldots, n\}$ denotes to the set of all input neurons N_i with terminal fibers that synapse on the kth dendrite of a morphological neuron $N, L \subseteq \{0, 1\}$ corresponds to the set of terminal fibers of the ith neuron that synapse on the kth dendrite of N, and $a_k \in \{-1, 1\}$ denotes the excitatory or inhibitory response of the kth dendrite.

Clearly, $I \neq 0$ y $L \neq 0$ since there is at least one axonal fiber coming from at least one of the input neurons with synapse dendrite k. The activation function used in a MNN is the hard limiter function that assigns 1 if the input is greater or equal to 0 and assigns 0 if the input is lesser than 0. A more detailed explanation can be found in [19,23].

3. The new training algorithm

A key issue in the design of a DMNN is its training; this is in the selection of the number of dendrites and the values of synaptic weights for each dendrite. For purposes of explaining the algorithm, a simple example of three classes with two attributes is used. Fig. 1 shows the example patterns, points of C^1 are shown as red dots, patterns of C^2 as green dots and points of C^3 as blue dots.

Given p classes of patterns, C^k, $k = 1, 2, \ldots, p$, each with n attributes, the algorithm applies the following steps:

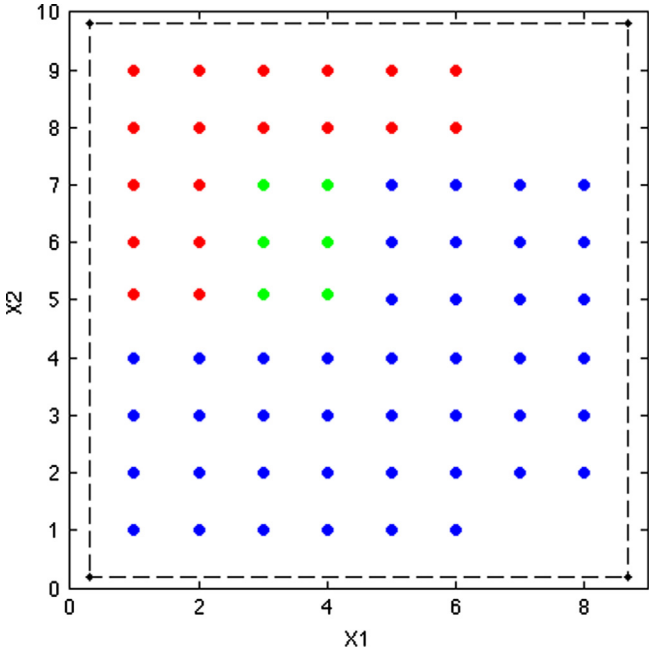

Fig. 1. Step 1: box that encloses all the sample patterns. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Step (1) Select the patterns of all the classes and open a hyper-cube HC^n (with n the number of attributes) with a size such that all the elements of the classes remain inside HC^n. The hyper-cube can be one whose coordinates match the patterns of class boundaries; it can be called the minimum hyper-cube *(MHC)*. For having better tolerance to noise at the classification time, add a margin M on each side of the *MHC*. This margin is a number greater or equal to zero and is estimated as a function of the size T of the *MHC*. If $M = 0.1T$ then the new hyper-cube will extend that ratio to every side of the *MHC*. Fig. 1 presents the box that covers all the patterns, with $M = 0.1T$.

Step (2) Divide the global hyper-cube into 2^n smaller hyper-cubes. Verify if each generated hyper-cube encloses patterns from only one class. If this is the case, label the hyper-cube with the name of the corresponding class, stop the learning process and proceed to step 4. For the example, the first division of the box is presented in Fig. 2.

Step (3) The step 3 has two stages: (a) if at least one of the generated hyper-cubes (HC^n) has patterns of more than one class, then divide HC^n into 2^n smaller hyper-cubes. Iteratively repeat the verification process onto each smaller hyper-cube until the stopping criterion is satisfied. Fig. 3 shows all the boxes generated by the training algorithm.

(b) Once all the hyper-cubes were generated, if two or more hyper-cubes of the same class share a common side, they are grouped into one region. Fig. 4 presents the application of this simplification procedure that automatically reduces the number of hyper-cubes.

Step (4) Based on the coordinates on each axis, calculate the weights for each hyper-cube that encloses patterns belonging to C^k. By taking into account only those hyper-cubes that enclose C^k items, at this moment the

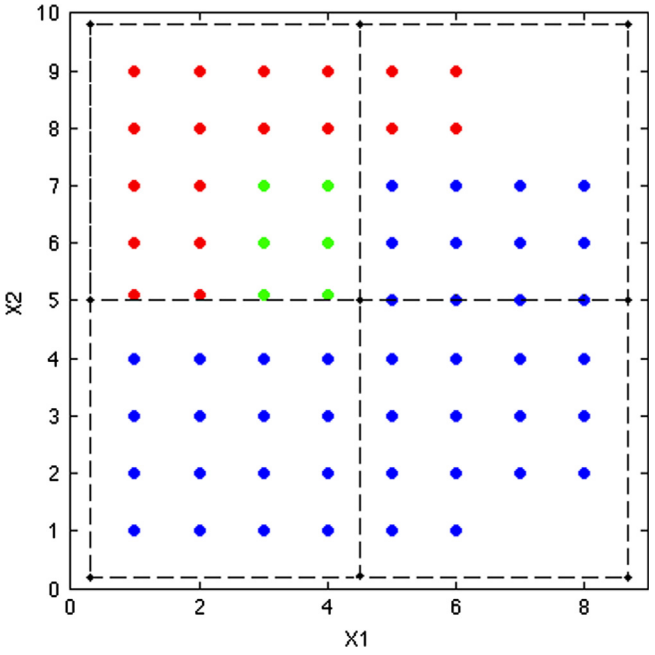

Fig. 2. Step 2: first division of the box executed by the training algorithm.

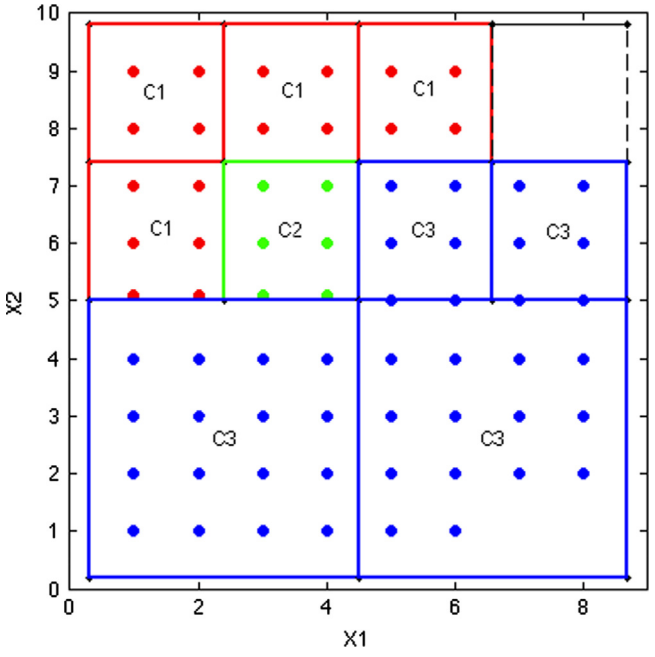

Fig. 3. Step 3 (a): boxes generated after the iterative division process.

DMNN is designed. Fig. 5 shows a dendrite morphological neural network with an input layer that separates the three classes: C^1, C^2 and C^3. The neurons of the input layer are connected to the next layer via the dendrites. The black and white circles denote excitatory and inhibitory connection, respectively. The geometrical interpretation of the computation performed by a dendrite is that every single dendrite determines a hyper-box which can be defined by a single dendrite via its weight values w_{ij} as the example shows.

For testing, two noisy patterns, $\tilde{X}_1 = \binom{4.5}{8.5}$ of the class C^1 and $\tilde{X}_2 = \binom{4}{3.5}$ of the class C^3, were selected as examples at this stage. When Eq. (3) is applied to the first dendrite, the following results

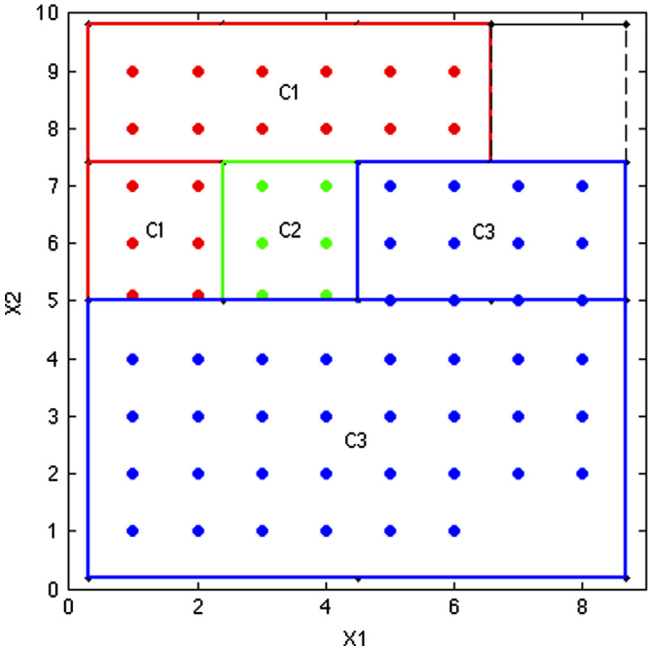

Fig. 4. Step 3 (b): boxes obtained after the simplification algorithm, points of class C^1 are enclosed by two red boxes, class C^2 is enclosed by one green box and C^3 by two blue boxes generated by the dendrites. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

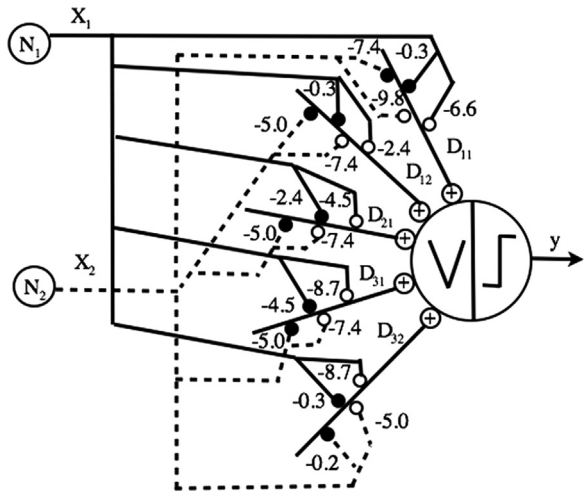

Fig. 5. Step 4: dendrite morphological neural network that solves the example problem.

are obtained:

$$D_{11}(\tilde{X}_1) = D_{11}\begin{pmatrix}4.5\\8.5\end{pmatrix} = [(4.5-0.3)\wedge -(4.5-6.6)]\wedge[(8.5-7.4)\wedge -(8.5-9.8)] = [2.1\wedge 1.1] = 1.1$$

$$D_{11}(\tilde{X}_1) = D_{11}\begin{pmatrix}4\\3.5\end{pmatrix} = [(4-0.3)\wedge -(4-6.6)]\wedge[(3.5-7.4)\wedge -(3.5-9.8)] = [2.6\wedge -3.9] = -3.9$$

In the same way, all the other dendrites are calculated; for $\tilde{X}_1$: $D_{12} = -2.1$, $D_{21} = -1.1$, $D_{31} = -1.1$, $D_{32} = -3.5$ and for $\tilde{X}_2$: $D_{12} = -1.6$, $D_{21} = -1.5$, $D_{31} = -1.5$, $D_{32} = 1.5$. With these values and by means of Eq. (1), the classification is obtained:

$$\tau(\tilde{X}_1) = (D_{11}\vee D_{12}\vee D_{21}\vee D_{31}\vee D_{32}) = (1.1\vee -2.1\vee -1.1\vee -1.1\vee -3.5) = 1.1.$$

$$\tau(\tilde{X}_2) = (D_{11} \vee D_{12} \vee D_{21} \vee D_{31} \vee D_{32})$$
$$= (-3.9 \vee -1.6 \vee -1.5 \vee -1.5 \vee 1.5) = 1.5.$$

Therefore $\tau(\tilde{X}_1) = 1.1 \geq 0$ corresponds to D_{11} (index of C^1) thus $y(\tilde{X}_1) = 1$, the input pattern is classified into class C^1, as was

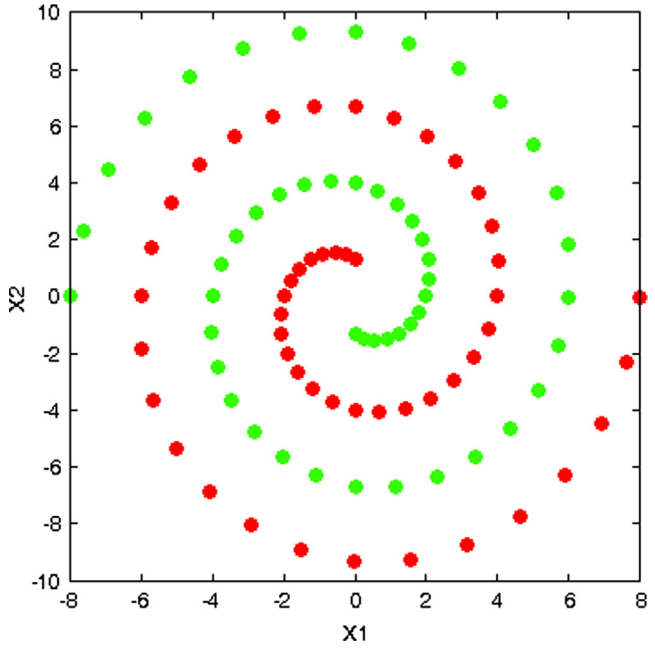

Fig. 6. Spirals synthetic training dataset with 50 samples.

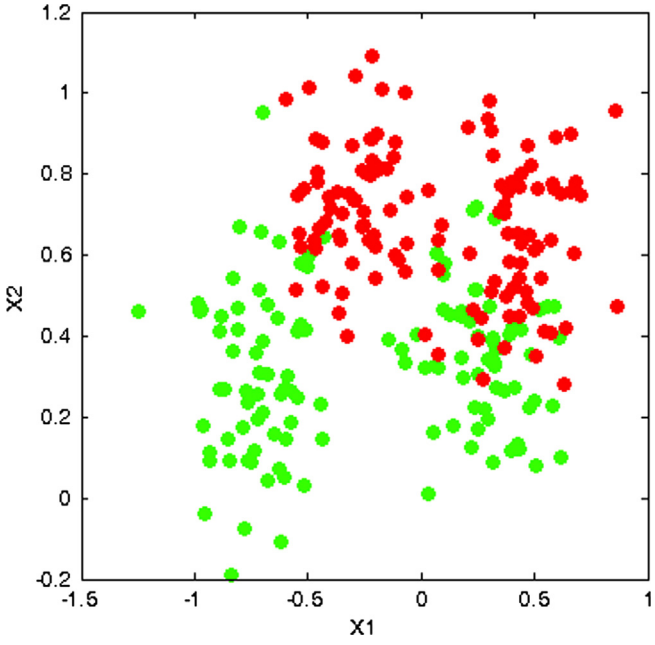

Fig. 7. Ripley's synthetic training dataset.

Table 1
Parameters values of the differential algorithm used to find the factor *M*.

Parameters	Value
Maximum number of generations	100
Population size	20
CR (crossover probability)	[0.5, 0.95]
F (differential weight)	[0.3,0.8]

Table 2
Comparison table of the classification error for the spiral problem obtained by the SLMP-PO, SLMP-P1 and SLMP-P2; where PO refers to the original proposal, P1 is the algorithm with *M* found by differential evolution and P2 is the algorithm with the maximum activation function and *M* obtained by the evolutive algorithm.

σ test	SLMP-PO			SLMP-P1			SLMP-P2		
	# Dendrites	*M*	% Error	# Dendrites	*M*	% Error	# Dendrites	*M*	% Error
0.2	28	0.3	4.4	28	0.22	3	34	0.25	0
0.3	28	0.3	6.44	28	0.06	4.03	30	0.27	1.17
0.4	28	0.3	11.31	28	0.63	7.54	45	0.21	3.87

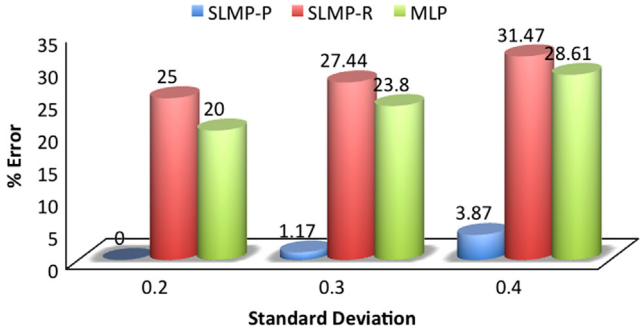

Fig. 8. Classification error comparison graphic of the MLP, SLMP-R and SLMP-P with different values of standard deviation of the test patterns.

Table 3
Comparison table of the MLP, SLMP-R and SLMP-P for the spiral problem.

σ test	MLP		SLMP-R		SLMP-P		
	# Neurons	% Error	# Dendrites	% Error	# Dendrites	*M*	% Error
0.2	9	20	11	25	34	0.25	0
0.3	9	23.8	12	27.44	30	0.27	1.17
0.4	9	28.61	14	31.47	45	0.21	3.87

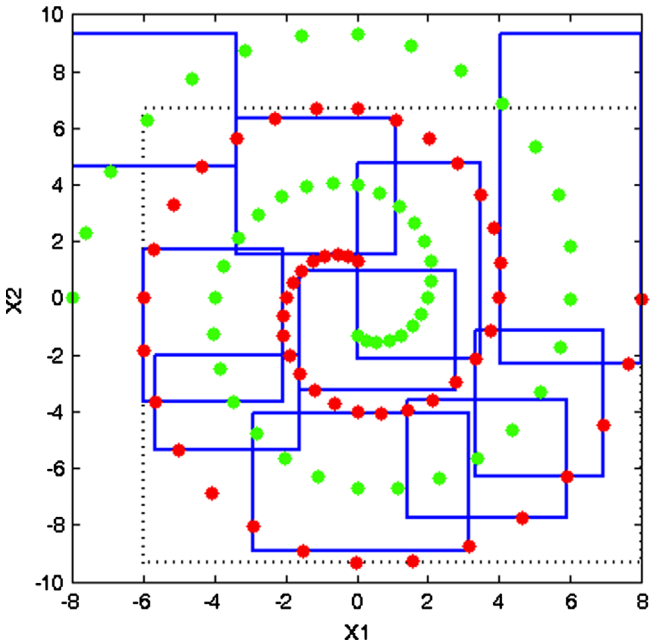

Fig. 9. Training result of the SLMP-R with 50 training samples.

expected. Moreover, $\tau(\tilde{X}_2) = 1.5 \geq 0$ corresponds to D_{32} (index of C^3) thus $y(\tilde{X}_2) = 3$, the input pattern is correctly classified into class C^3. If there is the case that the neuron value (τ) is not greater or equal to zero, then the pattern is not classified into any class.

4. Experimental results and discussion

In this section, validation experimental results are presented. These results demonstrate a superior learning performance of the proposed algorithm over the training algorithm for a DMNN proposed by Ritter. Furthermore, comparisons with Multilayer Neural Networks with one hidden layer, Support Vector Machines and Radial Basis Networks were made. The experiments were performed using synthetic 2-dimensional datasets and real datasets.

4.1. Experimental results using synthetic data

The training algorithm was applied to synthetic datasets, the first dataset was generated and forms two Archimedean spirals defined by the expressions

$$x_c(\theta) = \frac{2(-1)^{1-c}}{\pi}\theta \cos(\theta) \tag{4}$$

$$y_c(\theta) = \frac{8(-1)^{1-c}}{3\pi}\theta \sin(\theta). \tag{5}$$

where $c \in \{0, 1\}$ denotes the spiral class label, θ is the angle in radians, and (x_c, y_c) are the coordinates of the spiral points. A more detailed explanation can be found in [22].

Using the above formulas, samples for the training dataset were generated. Fig. 6 shows the spirals training dataset with 50 samples. In the following figures, solid dots represent the training points for the classes C^k. Empty circles denote the test samples and asterisks represent the test samples classified in each class.

The second dataset was the Ripley's synthetic dataset that consists of two classes [18]. The data is divided into a training dataset and a test set consisting of 250 and 1000 samples, respectively, with the same number of samples belonging to each of the two classes. The training dataset appears in Fig. 7.

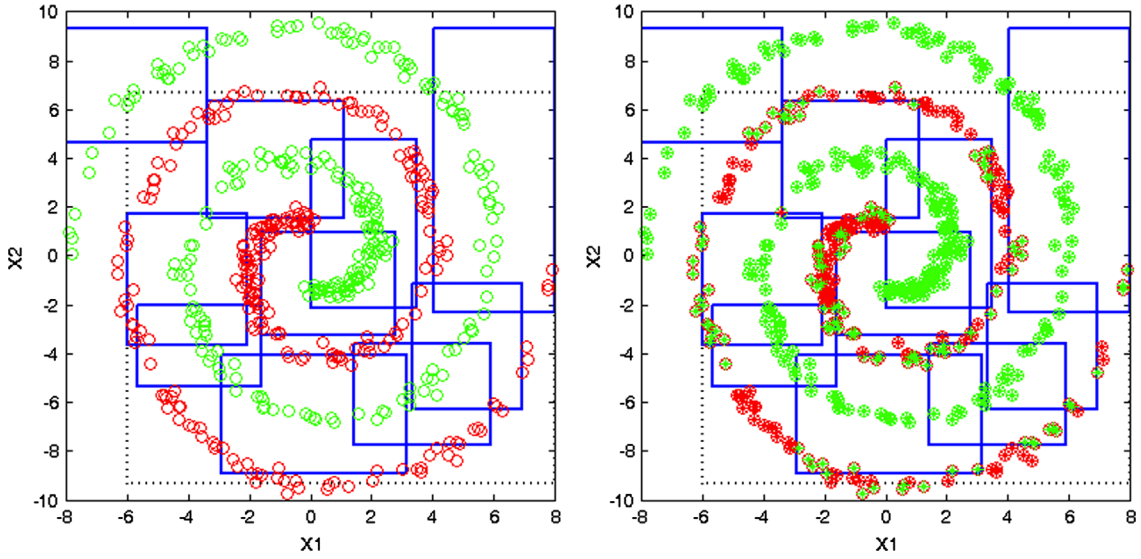

Fig. 10. Classification results of the SLMP-R with 50 training samples and 500 test samples.

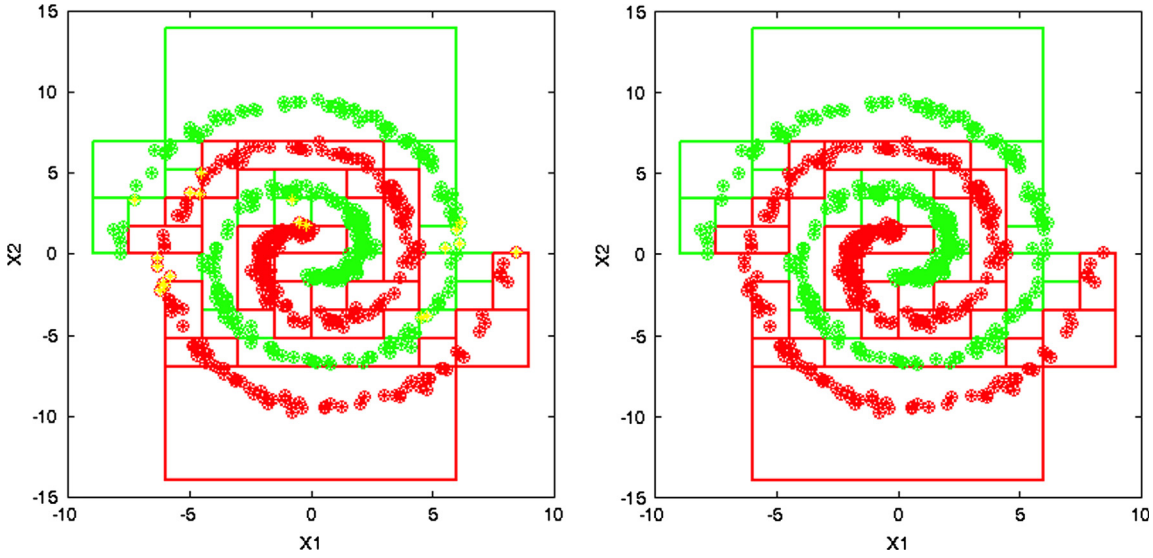

Fig. 11. Classification results of the SLMP-P1 (hard limiter) and SLMP-P2 (maximum function) with 50 training samples and 500 test samples. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

In the original proposed training algorithm for a single layer morphological perceptron (SLMP-PO) the value of factor M was selected empirically. In a second approach, the value of M that minimizes the error was determined by means of an evolutionary algorithm, differential evolution DE/rand/1/bin [16], with the parameters shown in Table 1.

To ensure that there is no sensitivity to "F" and "CR" parameters, these parameter values were generated randomly (using a uniform distribution) per run between the intervals shown in Table 1. The intervals for both parameters were defined empirically. As the results reveal (see Table 2), the application of the differential evolution algorithm to find the best value of M, helps significantly to decrease the classification error of the proposed algorithm.

Besides the proposal to find the value of M, another improvement was made. The original DMNN model uses a hard limiter as the activation function. This activation function was changed by the maximum function. This decreases the error into a considerable percentage (see Table 2). This occurs because in the proposed algorithm some testing patterns not allocated to any class might exist. With the use of the maximum function, those patterns are assigned to the closest class (see Fig. 11).

Table 2 presents the classification errors obtained with the original proposed algorithm and the improvements for the spiral problem. To get the test datasets, noise was added to the training spiral patterns with a normal distribution with mean on the center of the spiral point and different values of standard deviation (σ), every testing dataset has 500 samples. All the algorithms were implemented using MATLAB 7.11 on a desktop computer with Intel i7 2.2 GHz processor, with 8 GB in RAM. The classification errors obtained show the advantages of the improvements as can be seen in the following table.

To verify the efficiency of the final proposed training algorithm for the single layer morphological perceptron (SLMP-P), training algorithm proposed by Ritter (SLMP-R) [22] and a multilayer perceptron (MLP) were used for comparison. It should be mentioned that the algorithm used in the following experiments is the SLMP-P2 which is referred to as SLMP-P. In all cases, the classification errors obtained by the proposed algorithm were smaller than those obtained by the Ritter's algorithm and the MLP. The results obtained with the final improved algorithm are very satisfactory. The number of dendrites generated by the proposed algorithm is bigger than the number of dendrites of the Ritter training algorithm, this is because in the proposed algorithm the generated hyper-cubes cover all the classes.

Fig. 8 shows an error percentage comparison graphic of the three algorithms. This graphic shows the superior training performance of the proposed training algorithm over the SLMP-R and the MLP (Table 3).

For the MLP with one hidden layer, the training parameters were established as: learning rate=0.25, momentum=0.2, the activation function used was the logistic sigmoid. Hecht-Nielsen [11] suggested that an upper limit for the number of hidden layer neurons should be smaller than $2N+1$ (for a special activation function) where N is the number of input neurons. Thus, this limit was considered as a starting point (4 neurons) but in this problem the error obtained with 4 neurons was considerable (between 40% and 50%). It was found that with 9 neurons in the hidden layer the error was acceptable.

Figs. 9, 10, 12 and 13 show the boxes formed by the training process using 50 samples and the test classification results obtained by the SLMP-R and the SLMP-P, respectively. From here, all classification result graphs show the test patterns and the

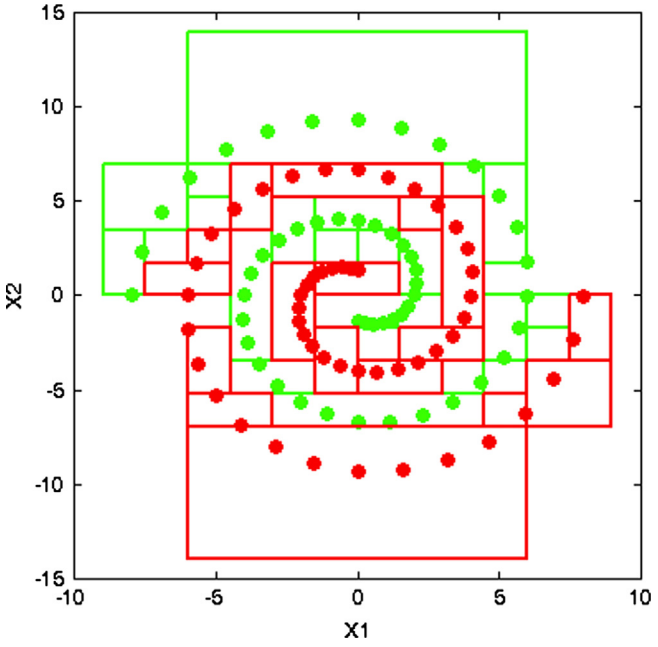

Fig. 12. Training result of the SLMP-P with 50 training samples.

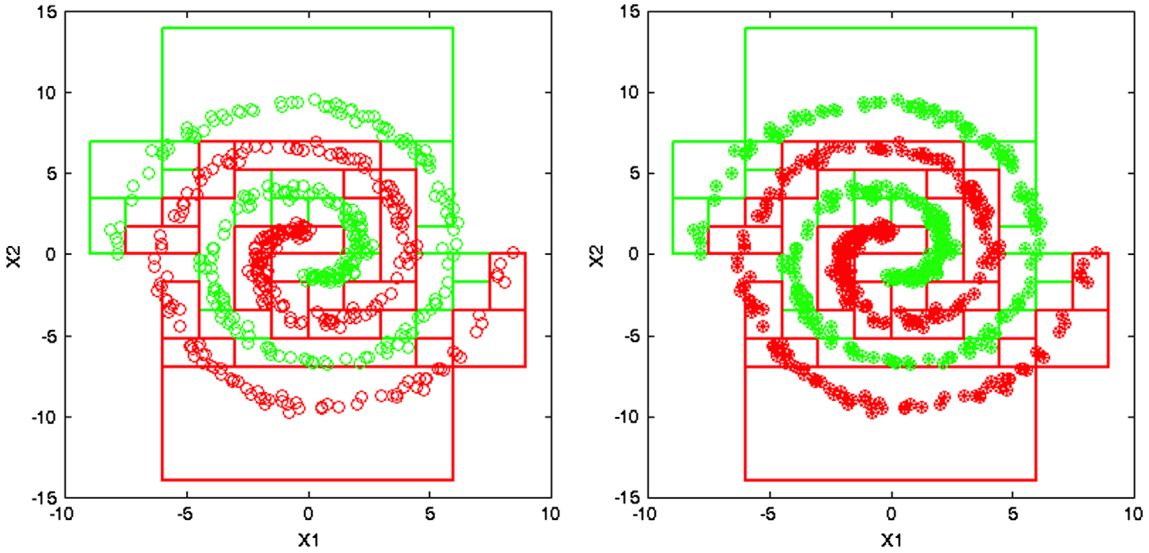

Fig. 13. Classification results of the SLMP-P with 50 training samples and 500 test samples.

training hyper-cubes to the right; to the left, the test samples and the classification results (asterisks) are presented.

Fig. 11 shows the results for the algorithm SLMP-P1 (hard limiter) to the right, in this case the patterns represented with yellow asterisks are not classified in any class, as can be seen there is an error. However, if the hard limiter activation function is changed by the maximum function, the error obtained in the spiral problem ($\sigma=0.2$) is zero and all the test patterns are assigned to one class.

Figs. 12 and 13 show the results with the SLMP-P for the spiral problem.

The classifications results of the algorithms for Ripleys dataset appear in Figs. 14, 15, 16 and 17.

Table 4 presents the classification errors for the Ripley's dataset obtained by each one of the algorithms. As can be seen, for this problem, the error obtained with the proposed training algorithm is larger that the error achieved in the spiral problem. This is because some samples of the data of C^1 are mixed in the class C^2 and vice versa. However, the SLMP-P improves the results of the SLMP-R and the MLP, so the performance of the proposed algorithm is satisfactory.

4.2. Experimental results using real data

In this subsection, the proposed training algorithm was applied to real data with p classes and n features for each class, and the results were compared with a Multilayer Perceptron (MLP), a Support Vector Machine (SVM) and a Radial Basis Network (RBN); the training algorithm proposed by Ritter can only be applied to the bi-class classification problems. First, object recognition experiments were considered and later some datasets from the UCI Machine Learning Repository [1] were used.

The proposed algorithm was tested to solve the 5 classes and 2 attributes problem presented in [33]. This problem consists in classifying the objects: sheave, milano tail, screw, eyebolt and spike.

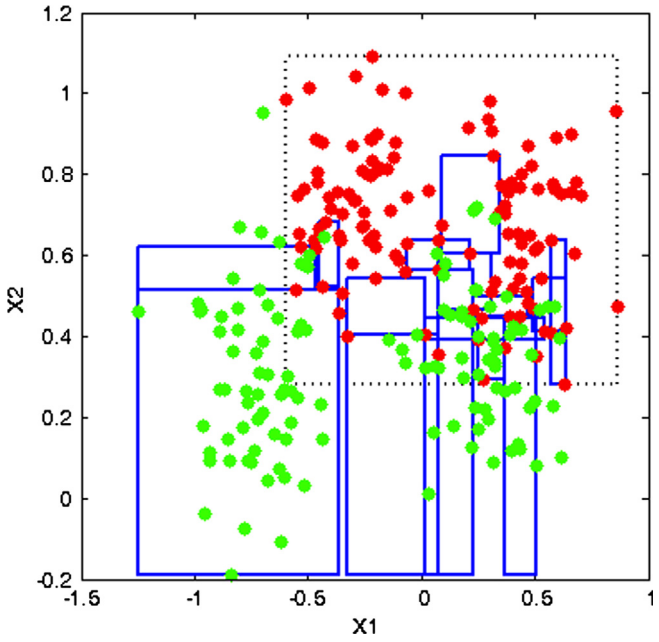

Fig. 14. Training result of the SLMP-R with 50 training samples.

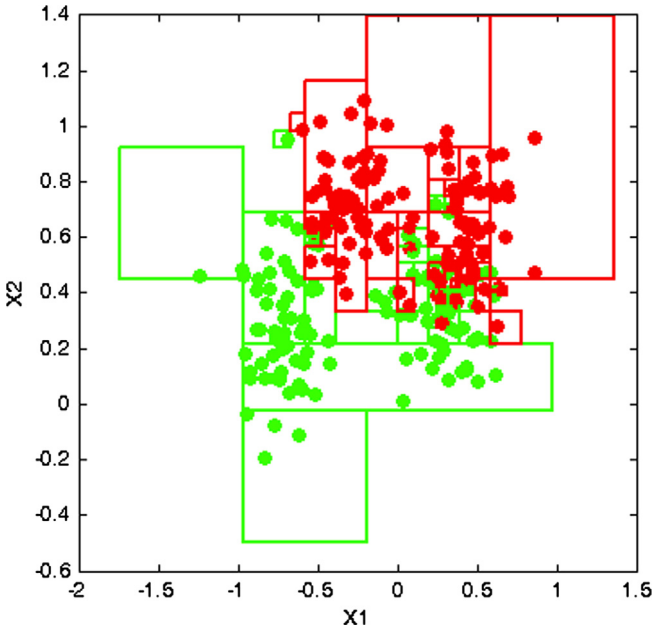

Fig. 16. Training result of the SLMP-P with 50 training samples.

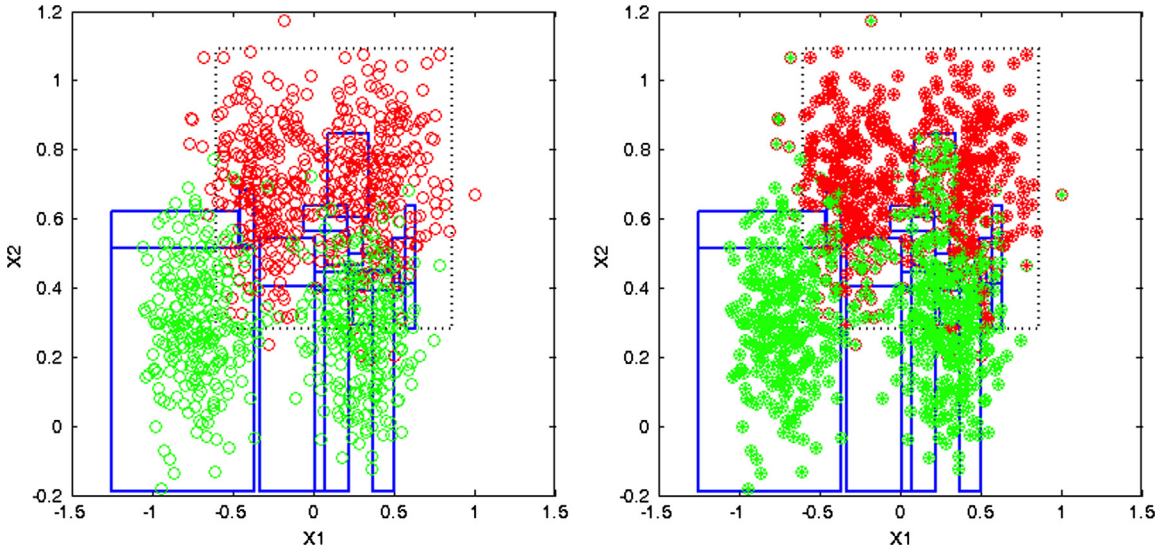

Fig. 15. Classification results of the 1000 test samples of the Ripley's dataset using SLMP-R.

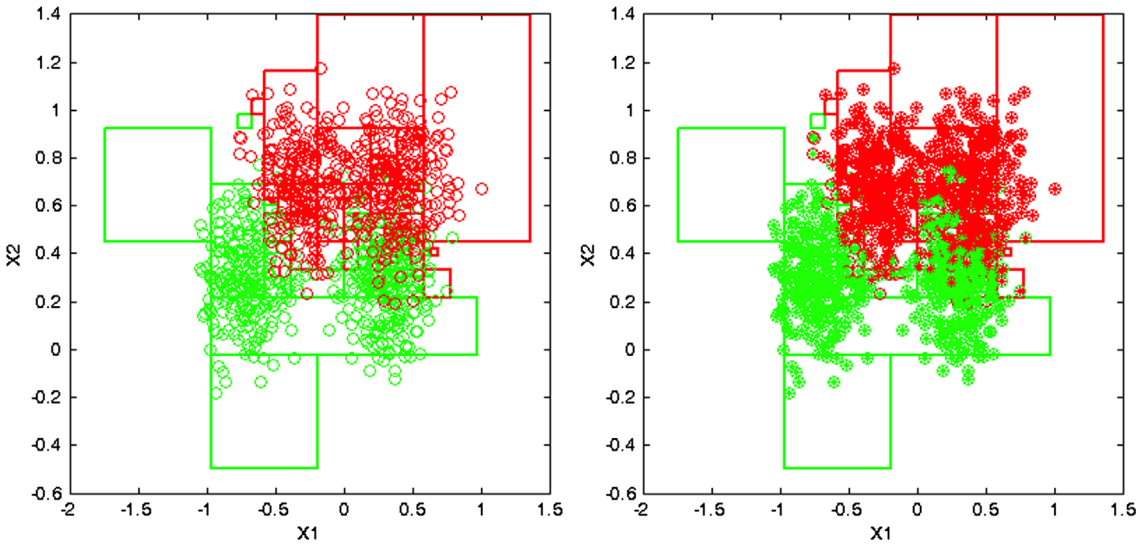

Fig. 17. Classification results of the 1000 test samples of the Ripley's dataset using SLMP-P.

Table 4
Comparison table of the MLP, SLMP-R and SLMP-P for the Ripley's dataset.

MLP		SLMP-R		SLMP-P		
# Neurons	% Error	# Dendrites	% Error	# Dendrites	M	% Error
9	14	16	18.8	70	0.238	12.2

The distinctive features used were the first and the second Hu moments. Fig. 18 shows the object images and their Hu moments.

The training and test datasets have 50 samples each. The proposed algorithm achieved a misclassification percentage of 0% with a value of $M=0.7$ and 6 dendrites. Fig. 19 shows the results.

To test the performance of the training algorithm in real images, a subset of the ETH80 database [14] was used. This dataset contains high-resolution color images of 80 objects from 8 different categories, the selected subset has 10 objects and each object is represented by 41 images from viewpoints spaced equally over the upper viewing hemisphere. The size of the images is of 128×128 pixels and the images were converted to grayscale. The 10 objects subset is shown in Fig. 20 and has the following objects: apple, car, cup-1, cup-2, dog-1, dog-2, pear-1, pear-2, tomato and cow.

For object representation, the first four Hu moments obtained from binary images were used. The results were compared with a one hidden layer MLP, a SVM and a RBF. These algorithms were applied using the software WEKA 3-6-9 [10]. For the MLP with one hidden layer, the training parameters were established as: learning rate=0.3 and momentum=0.2, the activation function used was the sigmoid. For the SVM, a Polynomial Kernel of degree 2 was used and for the RBN, two clusters were used. For selecting the Polynomial Kernel degree for the SVM and the clusters number for the RBN, these values were manually changed and the ones that generated the best results, were selected. The misclassification percentage obtained with the different neural networks are presented in Table 5.

Hence, as can be seen in Table 5, the modified DMNN improves the results of the remaining algorithms; however, the misclassification percentage could be reduced if more features to describe the objects are used; further research and experimentation is needed to verify this idea.

On the other hand, the classification performance of the DMNN was evaluated on seven well-known datasets, which can be found in the UCI Machine Learning Repository [1]. The Iris dataset was the first considered problem. This dataset has 3 classes (Iris setosa, Iris virginica and Iris versicolor) and 4 features (the length and the width of the sepals and petals, in centimeters). The dataset has 50 samples per class. The original dataset was divided into two subsets of the same size, taking samples randomly for training and testing.

To display the results, only three attributes were used (length and width of the sepals and length of the petals), in this way the patterns are in the 3D space and the results can be viewed graphically. Fig. 21 presents the result of the training and Fig. 22 shows the classification results. The algorithm achieved a 4% of misclassification error with $M=0.07$ and 22 dendrites.

The algorithm was applied to the Iris dataset with 4 attributes, the Glass Identification, the Liver Disorders, the Page Blocks, the Image Segmentation and Letter Recognition datasets [1]. The Glass Identification dataset used has 6 classes, 9 attributes and 214 samples (class 6 was not considered as it has very few samples). The Liver Disorders dataset has 2 classes, 6 attributes and 345 samples. The Page Blocks dataset has 5 classes, 10 attributes and 5473 samples. Like the Iris dataset, all the previous datasets were divided into two subsets to get the training and test patterns. The Image Segmentation dataset has 7 classes, 19 attributes and 210 samples for training and 2100 samples for testing. On the other hand, the Digits Handwriting Recognition dataset has 10 classes, 16 attributes and 10,992 samples with a training dataset of 7494 samples and a testing set of 3498 samples. The Letter Recognition dataset has 27 classes, 16 attributes and 20,000 samples, generally 16,000 samples are for training and 4000 for testing, in this work only 30% of the training and testing samples were used. For the Image segmentation and Letter recognition datasets, 10 attributes were used and 5 features were utilized for the Letter recognition dataset. For selecting the more useful attributes, the ranking algorithm "InfoGainAttributeEval" of the software WEKA was used. This algorithm evaluates the worth of an attribute by measuring the information gain with respect to the class. Table 6 presents the classification results for the different problems and a comparison with a MLP, a SVM and a RBN, with the parameters specified in the same way as for the ETH80 subset classification problem. As can be seen, in all cases the proposed algorithm obtains the lowest classification error. Furthermore, the proposed algorithm does

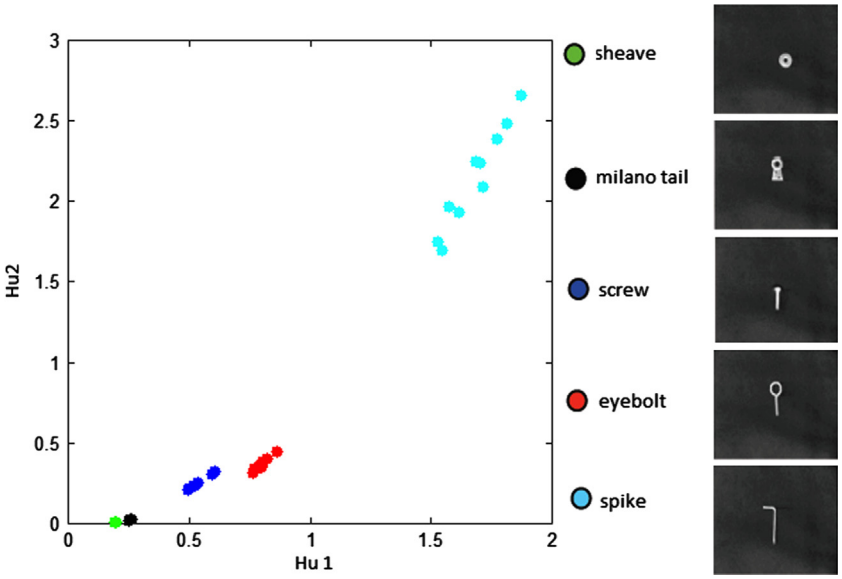

Fig. 18. Hu moments of the 5 classes.

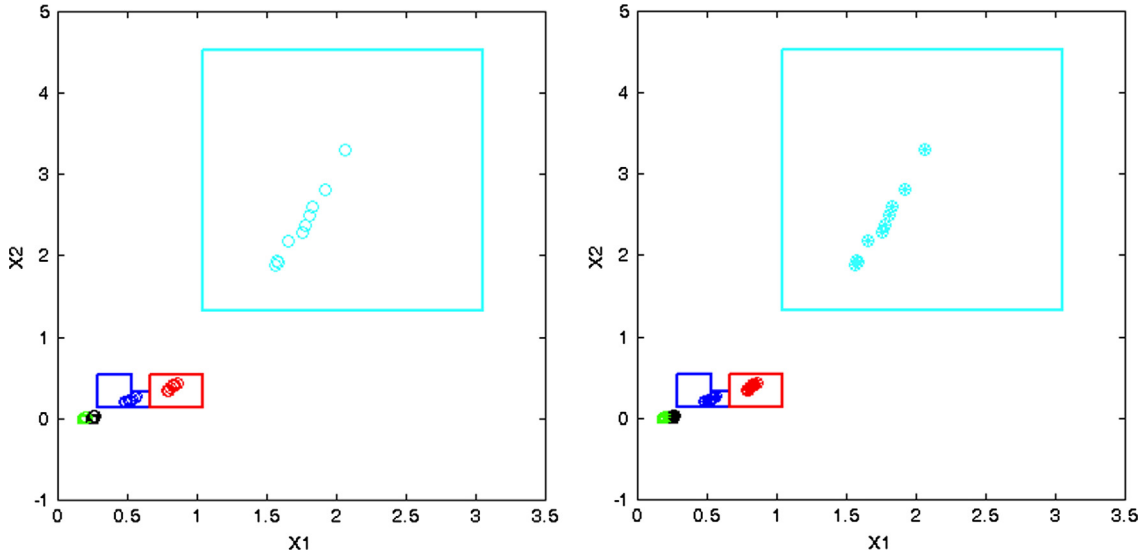

Fig. 19. Classification results of the object recognition problem.

Fig. 20. Subset of ETH80 database.

not manually set parameters so that the training process is very simple for the user.

The experiments reveal that the proposed algorithm has the following features:

1. Convergence in a finite number of steps.
2. Perfect classification of the training data.
3. No overlap between hyper-cubes with distinct class labels.
4. Once the value of M is set, if there are the same training patterns, the hyper-cubes generated are always the same.
5. No dependency of class presentation order.
6. No areas of indecision in the testing.
7. The algorithm can be applied to classification problems of p classes and n attributes.

Table 5
Comparison classification results table of the MLP, SVM, RBN and SLMP-P for the ETH80 subset.

MLP	SVM	RBN	SLMP-P		
% Error	% Error	% Error	# Dendrites	M	% Error
33.66	39.51	45.36	114	0.438	29.76

All these features make the proposed algorithm very efficient for solving different classification problems successfully. A main drawback of the proposed algorithm is that it grows exponentially as the number of describing dimensions increases. In the above experiments, the training time for the Iris dataset (4 features) was 0.04 s and was increased to 16.88 s for the Image segmentation dataset and to 172.58 s for the Digits Handwriting Recognition dataset, both sets with 10 attributes. However, due to the nature of the algorithm, it can be parallelized and the training time for problems with many attributes can be reduced.

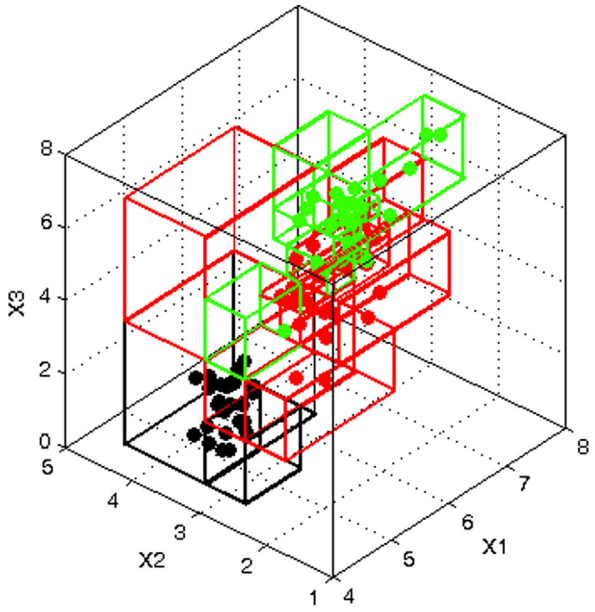

Fig. 21. Training result for the Iris problem with 3 features.

5. Conclusions

A novel and efficient training algorithm for a dendrite morphological neural network was presented. Comparisons of the proposed training algorithm with the SLMP-R, MLP, SVM and RBN were performed and the results demonstrated the advantages of the SLMP-P. The experiments revealed that the margin M included in the SLMP-P allows the algorithm to be less sensitive to noise than the SLMP-R and the value of M that minimizes the error can be found by an evolutive algorithm. Furthermore, the change of the hard limiter activation function, used in the original dendrite morphological neural network model, by the maximum function is very useful to reduce the error, because with this activation function, the testing patterns not allocated to any class are assigned to the closest class. Moreover, unlike the MLP, there are no convergence problems and always there is a perfect training data classification.

On the other hand, validation experimental results show that the training algorithm is used to solve real problems; however, to solve problems with many class attributes, the algorithm could be slow but this inconvenience could be reduced because due to its nature, the algorithm can be parallelized. Hence, future work will be focused on the implementation of the algorithm on a parallel architecture to handle multiclass problems with lots of describing

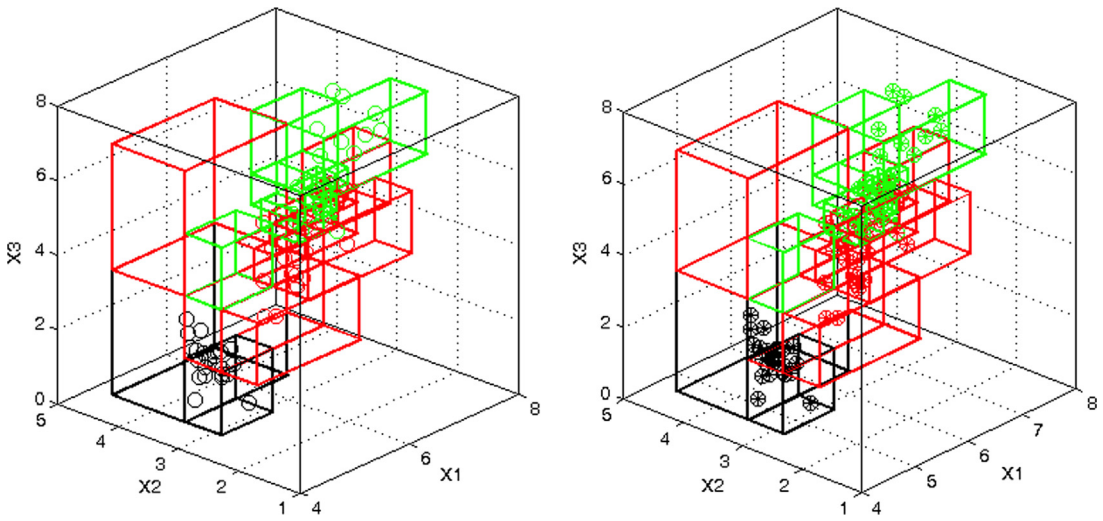

Fig. 22. Classification results for the Iris problem with 3 features.

Table 6
Comparison table of the MLP, SVM, RBN and SLMP-P for classification problems of p classes and n attributes.

Problem	MLP	SVM		RBN		SLMP-P		
	% Error	Degree	% Error	# Clusters	% Error	# Dendrites	M	% Error
Iris	6.77	1	4.00	2	5.33	20	0.30	2.67
Glass	31.46	5	31.68	6	31.68	64	0.19	30.69
Liver disorders	39.50	3	38.95	4	35.47	132	0.88	35.47
Page blocks	4.93	6	5.27	4	5.12	304	0.29	4.86
Image segmentation	27.33	1	26.71	2	21.71	92	0.54	24.14
Letter recognition	40.33	2	43.33	2	41.33	1439	0.72	38.75
Digits recognition	15.38	1	13.23	2	14.09	2259	0.25	9.55

features and in the algorithm application in problems like real time object recognition.

Acknowledgments

H. Sossa would like to thank SIP-IPN, CONACYT and ICYTDF under Grants 20121311, 20131182, 15014 and 325/2011 for the economical support to carry out this research. E. Guevara thanks CONACYT for the scholarship granted to pursue her doctoral studies.

References

[1] A. Asunción, D.J. Newman, UCI Machine Learning Repository, 2007.
[2] R. Barrón, H. Sossa, H. Cortés, Morphological Neural Networks with Dendrite Computation: A Geometrical Approach, Lecture Notes in Computer Science, vol. 2905, Springer-Verlag, 2003, pp. 588–595.
[3] S. Cheng-tian, W. Ke-yong, Image target detection using morphological neural network, in: International Conference on Computational Intelligence and Security, 2009, pp. 234–236.
[4] D. Chyzhyk, M. Graña, Optimal hyperbox shrinking in dendritic computing applied to Alzheimer's disease detection in: MRI, 6th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2011); Advances in Intelligent and Soft Computing, vol. 87, 2011, pp. 543–550.
[5] J.L. Davidson, F. Hummer, Morphology neural networks: an introduction with applications, Circuits Syst. Signal Process. 12 (2) (1993) 177–210.
[6] R. De Aráujo, F. Madeiro, R.P. De Sousa, L.F.C. Pessoa, Modular morphological neural network training via adaptive genetic algorithm for designing translation invariant operators, in: 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006), vol. 2, 2006, pp. 873–876.
[7] M. Graña, Lattice computing in hybrid intelligent systems, in: Proceedings of the 12th International Conference on Hybrid Intelligent Systems (HIS), 2012, pp. 1–5.
[8] M. Gra na, A.I. Gonzalez-Acuña, Learning parsimonious dendritic classifiers, Neuro 109 (2013) 3–8.
[9] M. Graña (Ed.), Special issue on: lattice computing and natural computing, Neurocomputing 72 (10–12) (2009) 2065–2066.
[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD Explor. (2009) 11.
[11] R. Hecht-Nielsen, Kolmogorov's mapping neural network existence theorem, in: IEEE First Annual International Conference on Neural Networks, vol. 3, 1987, pp. 11–13.
[12] V.G. Kaburlasos, S.E. Papadakis, G.A. Papakostas, Lattice computing extension of the FAM neural classifier for human facial expression recognition, IEEE Trans. Neural Networks Learn. Syst. 24 (2013) 1526–1538.
[13] V.G. Kaburlasos (Ed.), Special issue on: information engineering applications based on lattices, Inf. Sci. 181 (10) (2011) 1771–1773.
[14] B. Leibe, B. Schiele, Analyzing appearance and contour based methods for object categorization, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2003, pp. 409–415.
[15] Y. Nong, W. Hao, W. Changyong, L. Fanming, W. Lide, Morphological neural networks for automatic target detection by simulated annealing learning algorithm, Sci. China (Ser. F) 46 (2003) 262–278.
[16] K. Price, S. Rainer, J. Lampinen, Differential Evolution – A Practical Approach to Global Optimization, Springer, 1998.
[17] B. Raducanu, M. Graña, P. Sussner, Morphological neural networks for vision based self-localization, in: IEEE International Conference on Robotics and Automation, 2001, pp. 2059–2064.
[18] B.D. Ripley. Pattern Recognition and Neural Networks, Cambridge University Press, 1996.
[19] G.X. Ritter, L. Iancu, G. Urcid, Morphological perceptrons with dendritic structure, in: 12th IEEE International Conference in Fuzzy Systems (FUZZ 2003), vol. 2, 2003, pp. 1296–1301.
[20] G.X. Ritter, M.S. Schmalz, Learning in lattice neural networks that employ dendritic computing, in: IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, 2006, pp. 7–13.
[21] G.X. Ritter, P. Sussner, An introduction to morphological neural networks, in: Proceedings of the 13th International Conference on Pattern Recognition, vol. 4, 1996, pp. 709–717.
[22] G.X. Ritter, G. Urcid, Lattice algebra approach to single-neuron computation, IEEE Trans. Neural Networks 14 (2) (2003) 282–295.
[23] G.X. Ritter, G. Urcid, Learning in lattice neural networks that employ dendritic computing, Comput. Intell. Based on Lattice Theory 67 (2007) 25–44.
[24] G.X. Ritter, G. Urcid, Perfect recall from noisy input patterns with a dendritic lattice associative memory, in: Proceedings of the International Joint Conference on Neural Networks, 2011, pp. 503–510.
[25] Ch. Roberson, D.D. Dankel II, A morphological neural network approach to information retrieval, in: Twentieth International Florida Artificial Intelligence Research Society Conference, 2007, pp. 184–185.
[26] I. Segev, Dendritic processing, in: M.A. Arbib, The Handbook of Brain Theory and Neural Networks MIT Press, 1998, pp. 282–289.
[27] P. Sussner, Morphological perceptron learning, in: IEEE ISIC/CIRA/ISAS Joint Conference, 1998, pp. 477–482.
[28] P. Sussner, E.L. Esmi, Morphological perceptrons with competitive learning: lattice-theoretical framework and constructive learning algorithm, Inf. Sci. 181 (2011) 1929–1950.
[29] P. Sussner, C.R. Medeiros, An introduction to morphological associative memories in complete lattices and inf-semilattices, in: IEEE International Conference on Fuzzy Systems, 2012, pp. 1–8.
[30] G. Urcid, G.X. Ritter, J.C. Valdiviezo, Grayscale image recall from imperfect inputs with a two layer dendritic lattice associative memory, in: Third World Congress on Nature and Biologically Inspired Computing, 2012, pp. 261–266.
[31] G. Urcid, G.X. Ritter, J.C. Valdiviezo, Dendritic lattice associative memories for pattern classification, in: Fourth World Congress on Nature and Biologically Inspired Computing, 2012, pp. 181–187.
[32] M.E.R. Valle, M.A. Pimenta, Perceptrón morfológico de camada única (Technical Report), Department of Mathematics of the State University of Londrina, Brazil, 2005.
[33] R.A. Vázquez, H. Sossa, A new associative model with dynamical synapses, Neural Process. Lett. 28 (2008) 189–207.
[34] I. Villaverde, M. Graña, A. D'Anjou, Morphological neural networks for localization and mapping, in: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 2006), 2006, pp. 9–14.

Humberto Sossa was born in Guadalajara, Jalisco, Mexico in 1956. He received a B.Sc. Degree in Electronics from the University of Guadalajara in 1981, an M.Sc. in Electrical Engineering from CINVESTAV-IPN in 1987 and a Ph.D. in Informatics from the National Polytechnic Institute of Grenoble, France in 1992. He is a full time professor at the Centre for Computing Research of the National Polytechnic Institute of Mexico. His main research interests are in Pattern Recognition, Associative Memories, Image Analysis, and Robot Control using Image Analysis.

Elizabeth Guevara was born in Puebla, Puebla, Mexico in 1976. She received a B.Sc. Degree in Electronics from the University of Puebla in 2000 and an M.Sc. in Intelligent Systems from ITESM, Campus Monterrey in 2003. She is currently studying a Ph.D. in Computer Science at the Centre for Computing Research of the National Polytechnic Institute of México. Her main research interests are in Robotics, Computer Vision and Neural Networks.