Dendrite Morphological Neural Networks Trained by Differential Evolution

Fernando Arce[1], Erik Zamora[2], Humberto Sossa[1], Ricardo Barrón[1]
Instituto Politécnico Nacional
CIC[1], UPIITA[2]
Av. Luis Enrique Erro S/N, Unidad Profesional Adolfo López Mateos, C.P. 07738
Mexico City
Emails: b150689@sagitario.cic.ipn.mx, ezamorag@ipn.mx, hsossa@cic.ipn.mx, rbarron@cic.ipn.mx

***Abstract*—A new efficient training algorithm for a Dendrite Morphological Neural Network is proposed. Based on Differential Evolution, the method optimizes the number of dendrites and increases classification performance. This technique has two initialisation ways of learning parameters. The first selects all the patterns and opens a hyper-box per class with a length such that all the patterns of each class remain inside. The second generates clusters for each class by k-means++. After the initialisation, the algorithm divides each hyper-box and applies Differential Evolution to the resultant hyper-boxes to place them in the best position and the best size. Finally, the method selects the set of hyper-boxes that produced the least error from the least number. The new training method was tested with three synthetic and six real databases showing superiority over the state-of-the-art for Dendrite Morphological Neural Network training algorithms and a similar performance as well as a Multilayer Perceptron, a Support Vector Machine and a Radial Basis Network.**

I. Introduction

Linear perceptrons divide the input space into several regions, using hyper-planes as decision boundaries. In a case where more linear layers are connected, perceptrons divide the input space by a hyper-surface. In contrast, Morphological Neural Networks (MNN) divide the same space by several piecewise lines that together can create complex decision boundaries, allowing non-linear classification with only one neuron. This is not viable with a one-layer linear perceptron. The morphological processing involves min and max operations. These operators generate the piecewise boundaries for classification problems, and have the advantage of being implemented easily in logic devices.

This paper focuses on a specific type of Morphological Network which is called a Dendrite Morphological Neural Network (DMNN). A neuron has dendrites, and each represents a hyper-box in high dimensional space (and a box in 2D). These dendrites have played an important role in the proposal of training methods. The common approach is to enclose patterns with hyper-boxes and label each to the correct class. There are many training methods that have been published based on this approach. Most are heuristics and are not based on the optimization of learning parameters. In this paper, we show that a parameter optimization can be computed and can improve the classification performance of DMNN over heuristic-based and other popular machine learning methods.

Differential Evolution is distinguished as a powerful optimization technique [1], [2], [3]. This method is especially useful for problems with cost functions that are non-differentiable, non-continuous, non-linear, multi-dimensional or have many local minima. The main advantage is that the method does not require calculating analytic expressions such as gradients. Futhermore, the method is robust to find the global optimization in comparison with gradient-based methods. In this paper, we adapt the Differential Evolution to find the number of dendrites and their dendrite parameters for classification problems. This is useful because a DMNN has a cost function that is non-continuous due to the argmax operator at the neuron output.

The contributions of this paper are: 1) to the best of our knowledge, a DMNN is trained by an evolutionary algorithm for the first time; 2) our proposal outperforms the most effective heuristic-based and other popular machine learning methods; 3) the code for our approach is available at [4].

The following sections are organized as follows. Section 2 provides the previous work of MNN. Section 3 introduces the DMNN computational basis. Section 4 presents the proposed algorithm. Section 5 discusses the experimental results to assess the effectiveness of our proposal. Finally, in Section 6 we give our conclusions and proposed future work.

II. Previous Work

Morphological neurons were proposed by Davidson and Ritter in their seminal paper [5]; their computing capabilities were studied by Ritter and Sussner [6], [7]. Taking into account that information processing occurs also in dendrites, not only in cell body of neurons [8], the dendrite morphological neurons were proposed as an extension in [9]. A key issue is their training; we need to determine automatically the number of dendrites and the dendrite weight values. Several training approaches have been proposed [9], [10], [11], [12], [13], [14], [15], [16], [17]. And some real problems have been solved by using MNN [18] [19] [20] [21] [22].

Most of these methods use heuristics to determine the location of the hyper-boxes in the input space. In contrast, we propose making a search of those that were best classified based on an evolutionary approach. This is not the first time that parameters optimization has been proposed in the context of morphological neurons. Araujo [23] and Pessao et al. [24] investigated gradient-based optimization for certain architectures of morphological neurons.

The main advantage of optimization based on evolution is that we can do without analytically evaluating expressions, e.g. gradients, and also deal with discontinuities of the cost function.

III. Computational Basis for Dendrite DMNN

MNN base their computations on Lattice algebra. More detailed information can be found in [25], [26].

For this research paper, we used a total connected DMNN with a single output neuron. The computation performed τ_k^j by kth dendrite for the jth class can be expressed by (1):

$$\tau_k^j = \wedge_{i=1}^n (x_i + w_{ik}^1) \wedge -(x_i + w_{ik}^0), \qquad (1)$$

where x_i is an input vector and n is the vector dimensionality, $i \in I$ and $I \in \{\mathit{1,...,n}\}$ represents the set of all input neurons with terminal fibers that synapse on the kth dendrite. w_{ik}^0 and w_{ik}^1 are the synaptic weights that correspond to the set of terminal fibers of the ith neuron that synapse on the kth dendrite; w_{ik}^1 is the activation terminal fiber and w_{ik}^0 is the inhibition terminal fiber.

On the other hand, the output value of a DMNN with a single output neuron τ_j is the maximum argument calculated with (2):

$$\tau_j = argmax_k(\tau_k^j), \qquad (2)$$

where the $argmax_k$ function selects only one dendrite from all dendrites of the network, τ_k^j vector.

Fig. 1 shows the architecture of a DMNN with a single output neuron and an example of a hyper-box generated by its dendrite weights in 2D.

IV. The Proposed Training Algorithm

This section is dedicated to explain the new training algorithm steps. Before presentation, two definitions are provided:

Definition 1. Let $x_1, x_2, \ldots, x_m$ be a finite set of sample patterns, where each pattern $x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in \mathbb{R}^n$ for $i = 1, 2, \ldots, m$ is a n-dimensional vector. Furthermore, each pattern x_i belongs to only one C^j class, for $j = 1, 2, \ldots, p$, where $p > 1$ is a finite number.

Definition 2. A hyper-box HB^n is a n dimensional box that contains a finite set of patterns $x \in \mathbb{R}^n$. This HB^n defines the weights w_i^l, $l = \{0, 1\}$ where 1 denotes excitation and 0 inhibition.

Equation (3) defines a hyper-box:

$$HB_k^n = \{x \in R^n : w_{ik}^1 \leq x_i \leq w_{ik}^0, i = 1, \ldots, n\}, \qquad (3)$$

so the purpose of our training algorithm is to create and place a set of hyper-boxes $HB_k^n \in \mathbb{R}^{k \times 2n}$ for $k \in \{1, 2, \ldots, K\}$ that establish an optimum decision boundary between the classes with the least number.

Algorithm 1 Pseudo-Code of DE Applied to DMNN.

```
Begin
Generate initial population of solutions.
for $d = 1$ to $q$
    Repeat:
        For the entire population, calculate the fitness value.
        For each parent, select two solutions at random and the best parent.
        Create one offspring using DE operators.
        If the offspring is better than parents:
            Parent is replaced by the offspring.
    Until a stop condition is satisfied.
End
```

A. Training Algorithm

This sub-section provides the steps of the proposed algorithm which has two initialisation methods for the hyper-boxes.

- HBd initialisation:
  1) For each C^j class, select all the patterns and open a hyper-box per class, with a length such that all the patterns of each class remain inside. Sussner and Laureano [14] proposed a similar algorithm which keeps dividing them in order to avoid overlapping. The suggested algorithm does not care about the hyper-boxes overlapping.
  2) Divide each hyper-box into smaller hyper-boxes along the first axis on equal terms by a factor d, for $d \in \mathbb{Z}^+$ until q divisions.
- K-means++ initialisation:
  1) Generates Nq clusters for each C^j class by using the clustering algorithm k-means++ [27]. These clusters are transformed to hyper-boxes.
- Apply DE as [28] to the actual hyper-boxes.
- Select the best set of hyper-boxes that produced the least error. Algorithm 1 shows the pseudo-code of DE applied to a DMNN.

a) *Initialization*: The set of hyper-boxes conforms a $k \times n$ matrix called $ListH$. Each row from $ListH$ has the corners of a hyper-box and represents a chromosome. The initial population of parents is formed by multiplying t random vectors F per $ListH$, where t is the population size. The multiplication of one vector F per $ListH$ conforms a parent of the total population and each multiplication changes the original size and position of each hyper-box. In this case, DE has preliminary knowledge about the solution space.

b) *Fitness function*: Based on the corners of the generated hyper-boxes that enclose the patterns for the C^j classes, minimize the classification error which is calculated by (4) dividing the g number of misclassified patterns by the total number of the m patterns:

$$\%error = \frac{g}{m} \times 100, \qquad (4)$$

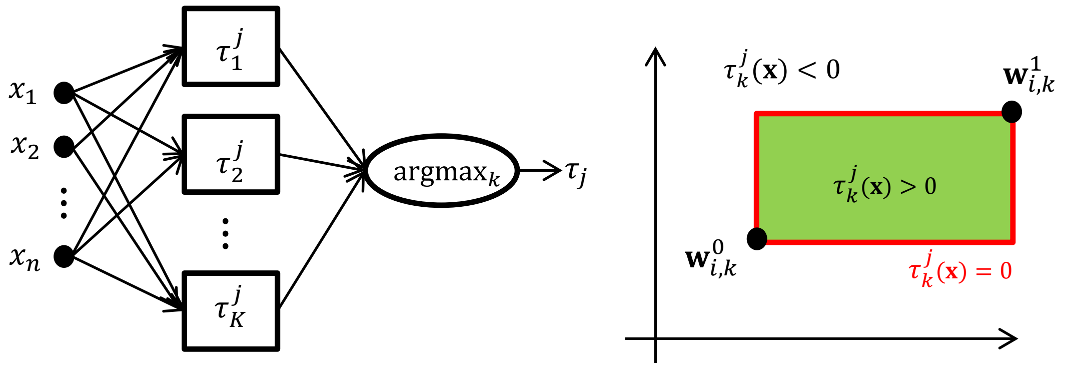


Fig. 1. Typical DMNN. Left: architecture of a DMNN with a single output neuron. Right: an example of a hyper-box in 2D generated by its dendrite weights.

c) *Mutation*: Select a parent, from each ten parents, and permute 30% of the k rows with the purpose of changing the class order.

d) *Crossover*: We do not report crossover due to obtaining similar results with and without it in the experimental section.

e) *Offspring creation using DE operators*: Create the offspring with the actual parents using a DE operator (5):

$$ListH_t = ListH_a + F \times (ListH_b - ListH_c), \qquad (5)$$

where a, b and c must be distinct. $ListH_t$ is the offspring, F is a random vector, $ListH_b$ and $ListH_c$ are randomly chosen parents and $ListH_a$ is the best member of the population.

f) *Stop condition*: DE stops if satisfying the error criteria or the q number of iterations.

B. Example of the Proposed Training Algorithm

One of the most important problems in the DMNN training algorithms is the election of the dendrite number and the optimization of the synaptic weights for each dendrite.

In order to explain this training algorithm, a straightforward example of three classes with two features is utilized, with the HBd initialisation method. Fig. 2 presents the problem to solve, blue dots belong to C^1, green dots belong to C^2 and red dots to C^3.

1) Select all the patterns of each class and open a hyper-box per class. Fig. 3 shows the hyper-boxes that covers all the patterns.

2) Divide each hyper-box by a factor d and apply DE to the resultant hyper-boxes in order to set them in the best position with the best size. Fig. 4 exhibits the divisions of each hyper-box generated in step 2 when $d = 1$, $d = 2$ and $d = 3$. When d is equal to 1 there is no division.

3) Select the best set of hyper-boxes. Fig. 5 displays the best generated after applying the training algorithm, with $d = 3$.

V. Experimental Results

Validation experimental results are presented in this section. The experiments were performed with three synthetic datasets and with six real datasets from the UCI Machine Learning

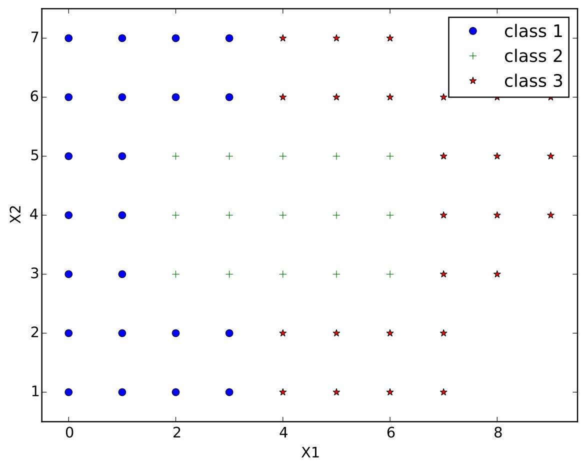


Fig. 2. Simple example of three classes with two attributes. Points of C^1 are shown as blue dots, points of C^2 as green dots and points of C^3 as red dots.

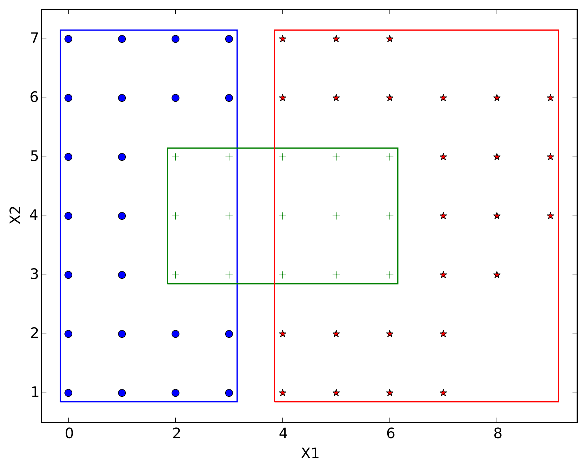


Fig. 3. Step 1. Enclosed patterns of each C^j class with a hyper-box per class. Blue HB^1 encloses all the patterns from C^1, green HB^2 encloses all the patterns from C^2 and the red HB^3 encloses all the patterns from C^3.

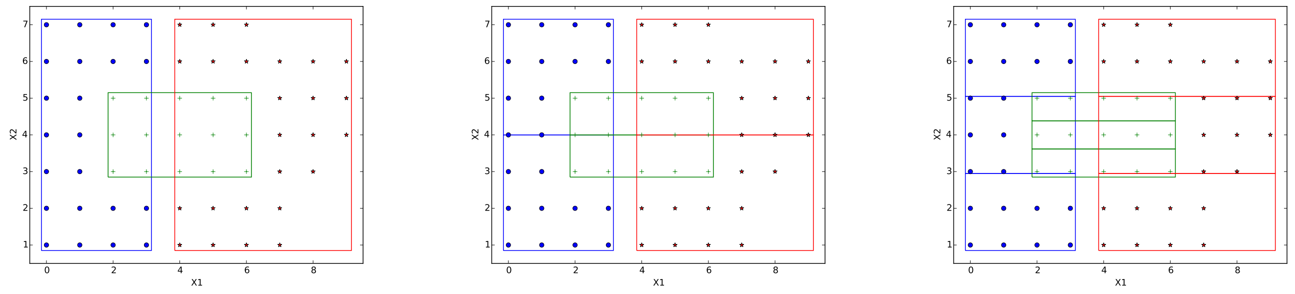


Fig. 4. Step 2. Divided each hyper-box by a factor d and apply DE to the resultant hyper-boxes. Left: when d is equal to 1, there is three hyper-boxes. Middle: when d is equal to 2, there is six hyper-boxes. Right: when d is equal to 3, there is nine hyper-boxes.

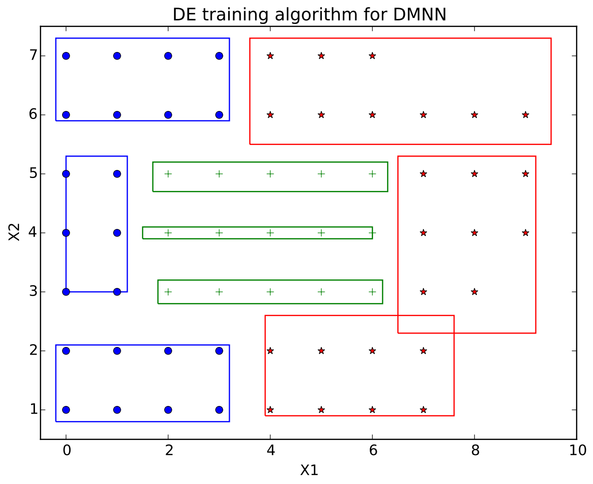


Fig. 5. Step 3. Selection of the best set of hyper-boxes generated by the training algorithm.

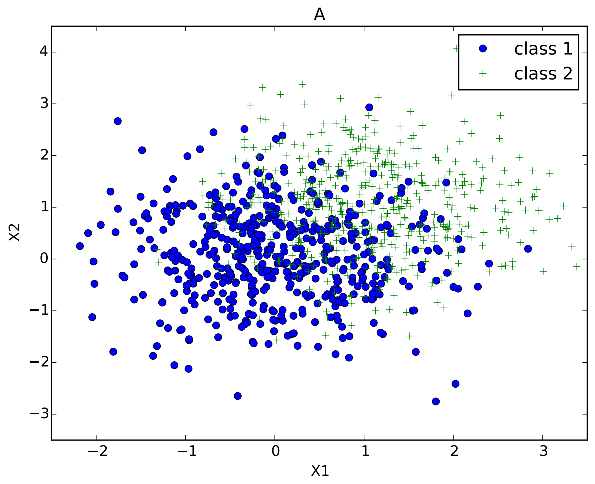


Fig. 6. Dataset “A” composed of two classes and two features, generated by two Gaussian distributions with a standard deviation equal to 0.9.

Repository [29]. On the basis of these results, this new training algorithm demonstrates a superior learning performance over the state-of-the-art for DMNN training algorithms and a considerable reduction of the dendrite number. Moreover, our training achieves similar performances to the most common classifers.

For the DE algorithm, and for all the datasets, the population size was set to 10 individuals with 50 generations (these parameters were chosen based on preliminary experiments).

A. Experimental Results Using Synthetic Data

In order to test the proposed training algorithm, it was first applied to two synthetic datasets with a high percent of overlapping where the overfitting problem can be more appreciable. It was compared with the algorithms SLMP-P [17] and SLMP-R [25] but mainly with the SLMP-P because it actually represents the state-of-the-art for DMNN training algorithms.

The first synthetic dataset was “A”, composed of two classes and two features, generated by two Gaussian distributions with a standard deviation equal to 0.9. The first class was centered around the point (0,0) and the second class around the point (1,1). Fig. 6 shows the dataset “A”.

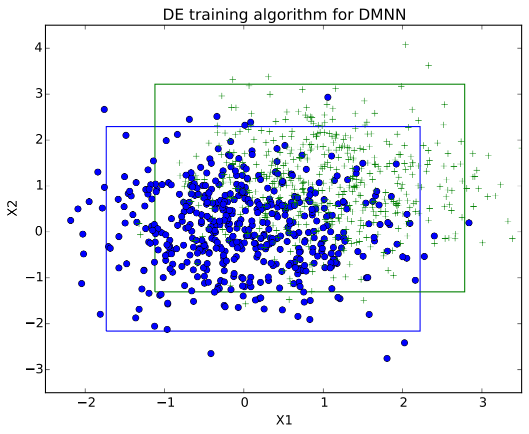
Fig. 7 presents the generation and placement of the hyper-boxes to solve the classification problem “A” with the DE for DMNN and SLMP-P training algorithms.

“B” was the second synthetic dataset and was formed by three Gaussian distributions with a standard deviation equal to 1.0. The classes were placed around the points (-1,-1), (1,1) and (-1,2). Fig. 8 exhibits the dataset “B”.

Fig. 9 shows the generations and the placement of the hyper-boxes to solve the classification problem “B” with the DE for DMNN and SLMP-P training algorithms.

The third synthetic dataset was “Spiral 2” which is a double spiral with two laps. Fig. 10 displays “Spiral 2” solved by the proposed algorithm; this dataset was the only one which used the K-means++ initialisation.

In these three synthetic datasets, 20% of data was used for testing and 80% for training. Table I presents the classification errors and the number of dendrites obtained with the new training algorithm and with the actual best DMNN training algorithms (the SLMP-P and SLMP-R algorithms). The reduc-



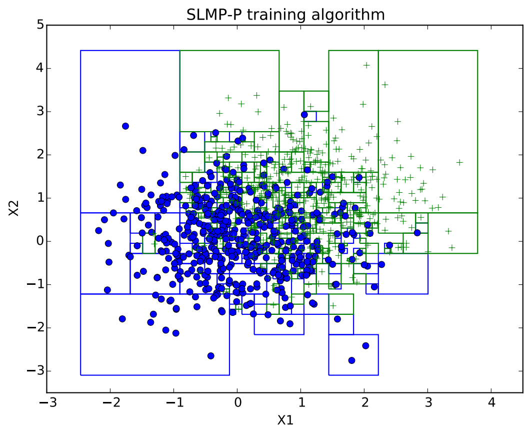


Fig. 7. Hyper-boxes generated by the DE for DMNN and SLMP-P training algorithms for problem "A". Left: 2 hyper-boxes generated with DE training algorithm for DMNN. Right: 419 hyper-boxes generated with SLMP-P training algorithm.

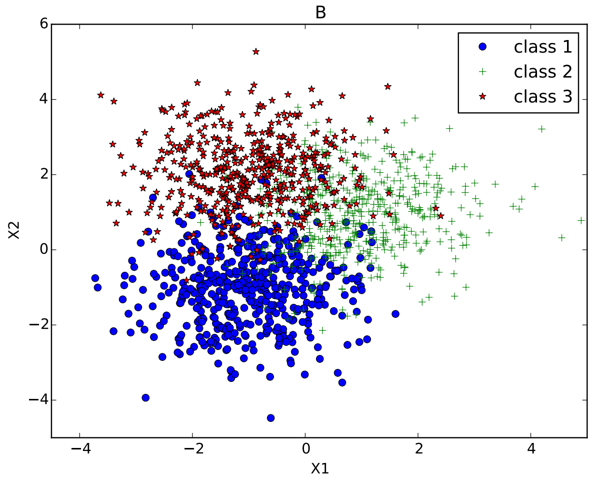


Fig. 8. Dataset "B" composed of three classes and two features, generated by three Gaussian distributions with a standard deviation equal to 1.0.

tion of the number of dendrites can be appreciated, and the classification error improvement obtained with the proposed algorithm.

B. Experimental Results Using Real Data

In this sub-section, the DE for DMNN was applied to real data (with p classes and n features) from the UCI Machine Learning Repository [29].

For the three synthetic and for the real datasets, the proposed algorithm was compared with the most common algorithms for classification: MLP [30], SVM [31] and RBN [32]. These final algorithms were applied using the software WEKA 3.6 and 3.8 [33] and the DMNN training algorithms were programed in Python 2.7.

In all the datasets, duplicated patterns were removed and the data was normalized; and all the classifiers were tuned-up. Table II shows the hyper-parameters used; where hL is the number of hidden layers, nu is the number of nerons in the hidden layers, LR is the learning rate, mo is the momentum, PD is the polynomial degree and nC is the number of clusters.

The well-known Iris dataset was the first considered problem which has 4 features (the length and the width of the sepal petals in centimeters), 3 classes (Iris setosa, Iris virginica and Iris versicolor) and 50 samples per class. For this dataset, the data was divided into 25% for testing and the rest for training. The others five datasets were Mammographic Mass, Liver Disorders, Glass Identification, Wine Quality and Mice Protein Expression.

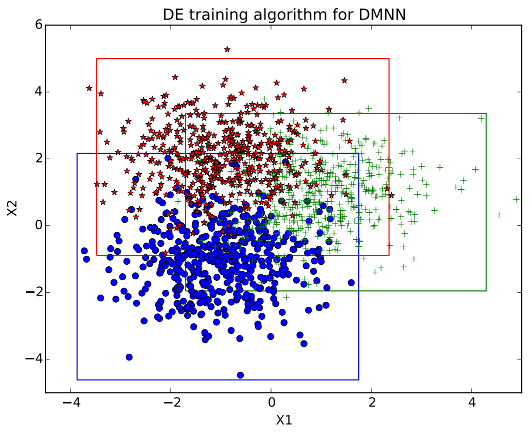
The Mammographic Mass dataset has 5 attributes (BI-RADS assessment, age, shape, margin and density) and 2 classes (benign and malignant); 20% testing, and 80% training. The third dataset was Liver Disorders which has 2 classes and 6 features. In this dataset, 15% of the data was used for testing and the rest for training.

Dataset Glass Identification has 10 features and 6 classes, 25% testing, and 75% training. The final two datasets were Wine Quality and Mice Protein Expression. Wine Quality consists of 11 features and 6 classes (25% testing, and 75% training) and for Mice Protein Expression 77 features and 8 classes were taken (20% testing and the rest for training).

Table III presents the results with DMNN training algorithms for the six UCI datasets. In all cases, the proposed algorithm achieves the smallest classification error and the number of dendrites is much less than the others. This allows us to implement this classifier in embedded devices to obtain real time responses.

Analyzing the results of the actual best training algorithm for DMNN (the SLMP-P [17]) with the proposed algorithm, it can be appreciated that the SLMP-P has the problem of overfitting. This is because the SLMP-P fits very well in the training set but fails to generalize in the testing set due to this technique being a heuristic and not an optimization method. For this reason, DE for DMNN overcomes the SLMP-P.

Table IV makes a comparison between the DE for DMNN



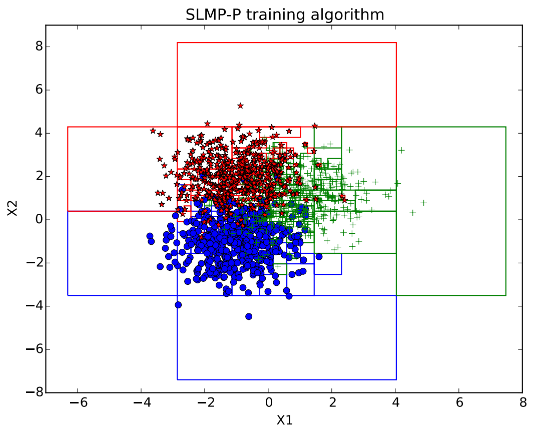


Fig. 9. Hyper-boxes generated by the DE for DMNN and SLMP-P training algorithms for problem "B". Left: 3 hyper-boxes generated with DE training algorithm for DMNN. Right: 505 hyper-boxes generated with SLMP-P training algorithm.

TABLE I
COMPARISON TABLE OF THE SLMP-R, SLMP-P AND DE FOR DMNN TRAINING ALGORITHMS FOR THE THREE SYNTHETIC DATASETS.

	SLMP-R		SLMP-P			DE for DMNN		
Dataset	$\#_{Dendrites}$	E_{test}	$\#_{Dendrites}$	E_{train}	E_{test}	$\#_{Dendrites}$	E_{train}	E_{test}
A	194	28.0	419	0.0	25.0	**2**	21.7	**20.5**
B	161	50.3	505	0.0	20.3	**3**	16.6	**15.2**
Spiral 2	160	8.6	356	0.0	7.2	**60**	7.3	**6.4**

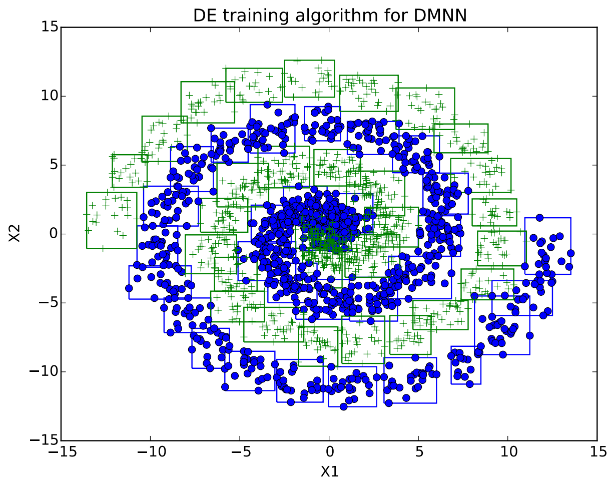


Fig. 10. "Spiral 2". Spiral of two laps solved by DE for DMNN with 60 hyper-boxes.

and the rest of the classifiers and shows that the DE training algorithm for DMNN not always achieved the least classification error, but was one of the best classifiers with the actual datasets.

In order to compare statistically the performance of the proposed algorithm with the previously mentioned classifiers (with a significance level $\alpha = 0.05$), Table V exhibits the obtained p-values in a paired t-test with the other methods.

TABLE II
HYPER-PARAMETERS USED FOR THE MLP, FOR THE SVM AND FOR THE RBN.

	MLP				SVM	RBN
Dataset	hL	nu	LR	mo	PD	nC
A	1	2	0.3	0.2	3	3
B	1	10	0.3	0.2	2	4
Spiral 2	1	22	0.3	0.7	3	6
Iris	1	5	0.3	0.2	1	2
Mammographic Mass	1	3	0.4	0.2	2	6
Liver Disorders	1	25	0.7	0.2	5	1
Glass Identification	2	2	0.2	0.1	1	4
Wine Quality	1	8	0.3	0.1	2	2
Mice Protein Expression	1	10	0.1	0.2	2	1

Based on these results, we can say that the DE for DMNN has a similar performance to the MLP, the SVM and the RBN, for the used datasets.

Finally, Table VI displays the results of applying 10-fold cross-validation to the nine datasets with DE for DMNN; where $best$ is the best result, $mean$ is the mean and std is the standard deviation of the folds for each dataset. DE for DMNN has a small standard deviation for most of the datasets (except for Mammographic, Liver Disorders and Glass Identification) which shows the method robustness.

VI. CONCLUSIONS AND FUTURE WORK

The research presented in this paper is an efficient and novel training algorithm for DMNN. Comparisons of the besought algorithm with the state-of-the-art for DMNN training algorithms demonstrated that DE for DMNN achieved the

TABLE III
COMPARISON TABLE OF THE SLMP-R, SLMP-P AND DE FOR DMNN TRAINING ALGORITHMS FOR THE SIX UCI DATASETS.

	SLMP-R		SLMP-P			DE for DMNN		
Dataset	$\#_{Dendrites}$	E_{test}	$\#_{Dendrites}$	E_{train}	E_{test}	$\#_{Dendrites}$	E_{train}	E_{test}
Iris	5	6.7	28	0.0	3.3	**3**	3.3	**0.0**
Mammographic Mass	51	14.4	26	0.0	19.2	**8**	15.8	**10.4**
Liver Disorders	41	42.0	183	0.0	35.5	**12**	37.6	**31.1**
Glass Identification	60	36.7	82	0.0	31.8	**12**	4.7	**13.6**
Wine Quality	120	51.0	841	0.0	42.1	**60**	42.1	**40.0**
Mice Protein Expression	77	18.9	809	0.0	5.0	**32**	6.6	**4.5**

TABLE IV
COMPARISON TABLE OF THE MLP, SVM, RBN AND DE FOR DMNN FOR ALL THE DATASETS.

	MLP		SVM		RBN		DE for DMNN	
Dataset	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
A	20.7	24.0	20.8	22.0	21.8	23.5	21.7	**20.5**
B	15.5	16.7	15.7	16.7	15.5	17.0	16.6	**15.2**
Spiral 2	6.6	7.4	45.1	44.4	47.4	45.2	7.3	**6.4**
Iris	1.7	**0.0**	4.2	**0.0**	4.2	**0.0**	3.3	**0.0**
Mammographic Mass	15.7	11.2	18.4	11.2	17.9	16.0	15.8	**10.4**
Liver Disorders	40.3	40.6	40.0	40.2	29.0	37.8	37.6	**31.1**
Glass Identification	14.1	20.4	12.3	18.2	0.0	20.4	4.7	**13.6**
Wine Quality	34.3	**39.0**	40.6	43.0	41.5	44.3	42.1	40.0
Mice Protein Expression	0.0	0.6	0.1	**0.5**	11.4	13.9	6.6	4.5

TABLE V
P-VALUES OF A PAIRED T-TEST TO COMPARE THE PERFORMANCE OF THE PROPOSED ALGORITHM WITH THE MLP, THE SVM AND THE RBN.

Classifiers	p-values
MLP	0.1713
SVM	0.1842
RBN	0.0618

TABLE VI
RESULTS OF APPLYING 10-FOLD CROSS-VALIDATION TO THE NINE DATASETS WITH DE FOR DMNN.

Dataset	*best*	*mean*	*std*
A	15.8	19.2	2.0
B	12.8	15.6	2.3
Spiral 2	3.6	7.7	2.5
Iris	0.0	1.3	2.7
Mammographic Mass	0.0	3.0	6.2
Liver Disorders	8.8	27.5	9.9
Glass Identification	0.0	6.4	6.5
Wine Quality	42.3	44.8	1.5
Mice Protein Expression	2.8	5.4	1.1

smallest classification error and the number of dendrites was much less than the other DMNN training algorithms for all the used datasets. Additionally, the DE for DMNN showed a competitive performance compared with the MLP, the SVM and the RBN classifiers especially in the datasets with low dimensionality.

Experimental results demonstrated that the proposed training algorithm helped to resolve real problems. Nevertheless, for a same classification problem, the algorithm not always achieved the same solution, and for datasets with big dimensionality the algorithm was time consuming. For this reason, future work will be the implementation of the algorithm on a Graphic Processing Unit (GPU) focusing on applications that require real time responses such as object classification and segmentation.

ACKNOWLEDGMENT

E. Zamora and H. Sossa would like to acknowledge the support provided by UPIITA-IPN and CIC-IPN to carry out this research. This work was economically supported by SIP-IPN (grants numbers 20160945 and 20161116), and CONACYT [grant number 155014 (Basic Research) and grant number 65 (Frontiers of Science)]. F. Arce acknowledges CONACYT for the scholarship granted to pursue his Ph.D. studies.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1008202821328

[2] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, ser. Natural Computing. Springer-Verlag, January 2005, iSBN 540209506.

[3] V. Feoktistov, *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[4] F. Arce, "Dendrite morphological neural networks trained by differential evolution - code," https://github.com/FernandoArce/, 2016.

[5] J. L. Davidson and G. X. Ritter, "Theory of morphological neural networks," pp. 378–388, 1990. [Online]. Available: http://dx.doi.org/10.1117/12.18085

[6] G. X. Ritter and P. Sussner, "An introduction to morphological neural networks," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 4, Aug 1996, pp. 709–717 vol.4.

[7] P. Sussner, "Morphological perceptron learning," in *Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings*, Sep 1998, pp. 477–482.

[8] I. Segev, "The handbook of brain theory and neural networks," M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, ch. Dendritic Processing, pp. 282–289.

[9] G. X. Ritter and G. Urcid, "Lattice algebra approach to single-neuron computation," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 282–295, Mar 2003.

[10] A. Barmpoutis and G. X. Ritter, “Orthonormal basis lattice neural networks,” in *Fuzzy Systems, 2006 IEEE International Conference on*, 2006, pp. 331–336.

[11] R. Barrón, H. Sossa, and H. Cortés, *Morphological neural networks with dendrite computation: A geometrical approach*. LNCS 2905, Springer-Verlag, 2003, pp. 588–595.

[12] G. X. Ritter and G. Urcid, *Computational Intelligence Based on Lattice Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. Learning in Lattice Neural Networks that Employ Dendritic Computing, pp. 25–44.

[13] D. Chyzhyk and M. Graña, *Optimal Hyperbox Shrinking in Dendritic Computing Applied to Alzheimer's Disease Detection in MRI*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 543–550.

[14] P. Sussner and E. L. Esmi, “Morphological perceptrons with competitive learning: Lattice-theoretical framework and constructive learning algorithm,” *Information Sciences*, vol. 181, no. 10, pp. 1929 – 1950, 2011, special Issue on Information Engineering Applications Based on Lattices. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025510001283

[15] H. Sossa and E. Guevara, *Modified Dendrite Morphological Neural Network Applied to 3D Object Recognition*. LNCS 7914, Springer-Verlag, 2013, pp. 314–324.

[16] G. X. Ritter, G. Urcid, and V. N. Juan-Carlos, “Two lattice metrics dendritic computing for pattern recognition,” in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, July 2014, pp. 45–52.

[17] H. Sossa and E. Guevara, “Efficient training for dendrite morphological neural networks,” *Neurocomputing*, vol. 131, pp. 132 – 142, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213010916

[18] ——, *Modified Dendrite Morphological Neural Network Applied to 3D Object Recognition on RGB-D Data*. LNAI 8073, Springer-Verlag, 2013, pp. 304–313.

[19] R. Vega, E. Guevara, L. E. Falcon, G. Sanchez-Ante, and H. Sossa, *Blood Vessel Segmentation in Retinal Images Using Lattice Neural Networks*. LNAI 8265, Springer-Verlag, 2013, pp. 529–540.

[20] L. Ojeda, R. Vega, L. E. Falcon, G. Sanchez-Ante, H. Sossa, and J. M. Antelis, *Classification of hand movements from non-invasive brain signals using lattice neural networks with dendritic processing*. LNCS 9116, Springer-Verlag, 2015, pp. 23–32.

[21] B. Gudiño, H. Sossa, G. Sanchez, and J. M. Antelis, *Classification of Motor States from Brain Rhythms Using Lattice Neural Networks*. LNCS 9703, Springer-Verlag, 2016, pp. 303–312.

[22] R. Vega, G. Sanchez, L. E. Falcon, H. Sossa, and E. Guevara, “Retinal vessel extraction using lattice neural networks with dendritic processing,” *Computers in Biology and Medicine 2015, 58:20-30. DOI: 10.1016/j.compbiomed.2014.12.016*.

[23] R. de A. Araujo, “A morphological perceptron with gradient-based learning for brazilian stock market forecasting,” *Neural Networks*, vol. 28, pp. 61 – 81, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608011003200

[24] L. F. Pessoa and P. Maragos, “Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition,” *Pattern Recognition*, vol. 33, no. 6, pp. 945 – 960, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320399001570

[25] G. X. Ritter, L. Iancu, and G. Urcid, “Morphological perceptrons with dendritic structure,” in *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on*, vol. 2, May 2003, pp. 1296–1301 vol.2.

[26] G. X. Ritter and M. S. Schmalz, “Learning in lattice neural networks that employ dendritic computing,” in *Fuzzy Systems, 2006 IEEE International Conference on*, 2006, pp. 7–13.

[27] D. Arthur and S. Vassilvitskii, “K-means++: The advantages of careful seeding,” in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[28] D. Ardia, K. Boudt, P. Carl, K. M. Mullen, and B. G. Peterson, “Differential Evolution with DEoptim: An application to non-convex portfolio optimization,” *The R Journal*, vol. 3, no. 1, pp. 27–34, 2011.

[29] D. N. A. Asuncion, “UCI machine learning repository,” 2007.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1,” D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362.

[31] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[32] D. S. Broomhead and D. Lowe, “Multivariable Functional Interpolation and Adaptive Networks,” *Complex Systems 2*, pp. 321–355, 1988.

[33] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: An update,” 2009.