Dendrite Morphological Neurons Trained by Stochastic Gradient Descent

Erik Zamora
Instituto Politécnico Nacional, UPIITA
Av. Instituto Politécnico Nacional 2580
Col. Barrio la Laguna Ticoman
Ciudad de México, 07340
Email: ezamorag@ipn.mx

Humberto Sossa
Instituto Politécnico Nacional, CIC
Av. Juan de Dios Batiz S/N
Col. Nueva Industrial Vallejo
Ciudad de México, 07738
Email: hsossa@cic.ipn.mx

***Abstract*—Dendrite morphological neurons are a type of artificial neural network that work with min and max operators instead of algebraic products. These morphological operators allow each dendrite to build a hyper-box in classification N-dimensional space. In contrast with classical perceptrons, these simple geometrical representations, hyper-boxes, allow the proposal of training methods based on heuristics without using of an optimisation method. In literature, it has been claimed that these heuristics-based trainings have advantages: there are no convergence problems, perfect classification can always be reached and training is performed in only one epoch. This paper shows that these assumed advantages come with a cost: these heuristics increase classification errors in the test set because they are not optimal and learning generalisation is poor. To solve these problems, we introduce a novel method to train dendrite morphological neurons based on stochastic gradient descent for classification tasks, using these heuristics just for initialisation of learning parameters. We add a softmax layer to the neural architecture for calculating gradients and also propose and evaluate four different methods to initialise the dendrite parameters. Experiments are performed based on several real and synthetic datasets. Results show that we can enhance the testing accuracy in comparison with solely heuristics-based training methods. This approach reaches competitive performance with respect to other popular machine learning algorithms. Our code developed in Matlab is available online.**

I. Introduction

Morphological neural networks are an alternative way to model pattern classes. Classical perceptrons divide the input space into several regions, using hyper-planes as decision boundaries. In a case where more layers are presented, perceptrons divide the input space using a hyper-surface. In contrast, morphological neurons divide the same space by several piecewise lines that together can create complex non-linear decision boundaries, allowing separation of classes with only one neuron. This is not possible with a one-layer perceptron because it is a linear model. The morphological processing involves min and max operations instead of multiplications. These comparison operators generate the piecewise boundaries for classification problems, and have the advantage of being implemented easier in logic devices than classical perceptrons.

This paper focuses on a specific type of morphological neuron which is called a Dendrite Morphological Neuron (DMN). This neuron has dendrites, and each dendrite can be seen as a hyper-box in high dimensional space (and a box in 2D). These hyper-boxes 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 right class. There are many training methods have been reported to exploit this approach. However, most of them are heuristics and are not based on the optimisation of learning parameters. Heuristics can be useful when numerical optimisation is computationally very expensive or impossible to compute. In this paper, we show this is not the case. A learning parameters optimisation can be computed in a reasonable time and can improve the classification performance of DMN over solely heuristics-based methods.

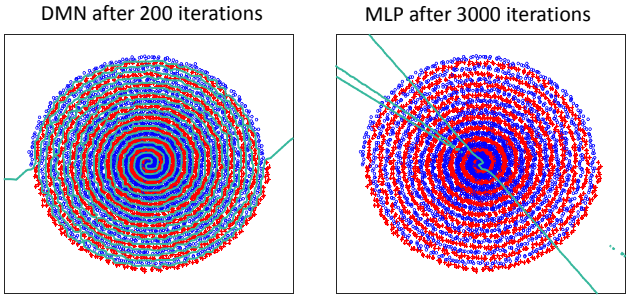

Fig. 1. We illustrate a classification problem where DMN outperforms a two-layer perceptron network (MLP). The decision boundaries are in blue-green (solid line). Although a double spiral with 10 laps follows a simple mathematical pattern, MLP cannot separate these classes. Both models were trained by SGD, but learning parameter initialisation is the key advantage for DMN.

The most important training method in the scope of classical perceptrons has been gradient descent [1]. Even though morphological networks have more than 25 years of development and backpropagation is the most successful training algorithm, most of the training methods for morphological neurons are based on intuition instead of on gradient descent. The extension of this approach is not trivial due to the problem of non-differentiability of morphological operations. We deal with this problem by adding a softmax layer [2], calculate the gradients and evaluate four different methods to initialise

978-1-5090-4240-1/16/$31.00 ©2016 IEEE

dendrite parameters. Fig. 1 shows a simple example where our proposal outperfoms classical multilayer perceptrons, using stochastic gradient descent (SGD) to train both models.

The contributions of this paper are:

- To the best of our knowledge, for the first time, a DMN architecture is extended with a softmax layer to be trained by SGD for classification tasks.
- We propose and evaluate four different methods based on heuristics to initialise dendrite parameters before training by SGD.
- The code for our approach is available at [3].

The rest of the paper is organised as follows. Section 2 provides a review of morphological neural networks. Section 3 introduces our approach to training DMN based on SGD. We present the experiments and results, as well as a short discussion in section 4 to show the effectiveness of our proposal. Finally, section 5 gives our conclusions and directions for future research.

II. Previous Work

Davidson and Ritter proposed the morphological neurons in the seminal paper [4] for template learning in dilation image processing. Sussner and Ritter studied their computing capabilities in [5], [6]. The DMNs were proposed as an extension in [7], taking into account that information processing occurs also in dendrites. Each dendrite represents a hyper-box in classification space. Improved neural architectures have also been proposed, using rotated hyper-boxes [8], hyper-boxes with softened corners [9] and the argmax operator as an activation function [10], [11], [12]. This last creates more complex non-linear boundaries and is used in this paper. In Fig. 2, we show the different types of decision boundaries that morphological neural networks can build to classify patterns.

A key issue of DMNs is training. We need to determine automatically the number of dendrites and the dendrite weight values. Several training approaches have been proposed [7], [14], [15], [16], [10], [12]. Most of them are based on heuristics to manipulate hyper-boxes. Only two proposals use optimisation to train morphological neural networks. 1) Morphological/rank/linear neural networks (MRL-NNs) [13] combine classical perceptrons with morphological/rank neurons. The training is based on gradient descent. They can solve complex classification problems such as recognising digits in images in shorter training times than MLPs. 2) Increasing morphological perceptron (IMP) [17] is a neural model similar to DMN but with a linear activation function. This has been applied to regression problems and is also trained by gradient descent. The drawback of these two training methods is that the number of hidden units must be tuned as a hyper-parameter. The advantage of creating more units during training is lost. Our approach in this paper preserves this advantage using heuristics-based training methods as pre-training. Particularly, our work differs from IMP in three ways: 1) we focus on classification problems instead of regression problems, 2) we reuse some heuristics to initialise hyper-boxes before training and 3) we extend the neural architecture using a softmax layer. Futhermore, our approach differs from MRL-NNs in the neural architecture, incorporating only morphological layers (no linear layers); and MRL-NNs uses no hyper-boxes which is a great difference because hyper-boxes make it easier to initialise learning parameters for gradient optimisation.

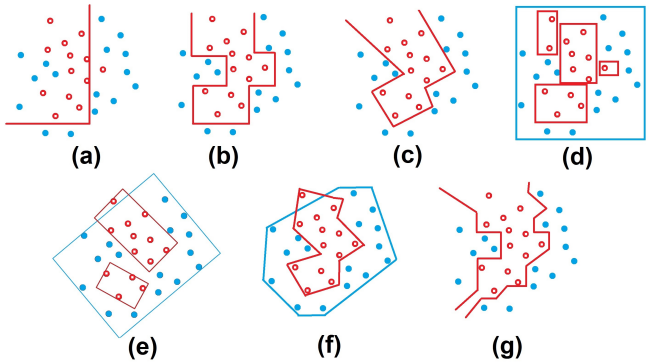

Fig. 2. We show the types of decision boundary for different morphological neural networks. (a) Single-Layer MP [5] uses semi-infinite hyper-boxes that are parallel to Cartesian axes. (b) Two-Layer MP [6] separates classes by hyper-planes parallel to Cartesian axes. (c) Multilayer MRL-NNs [13] classify data by general hyper-planes. Note that the decision boundaries for the next three networks are closed surfaces. (d) SLMP [7] uses hyper-boxes parallel to Cartesian axes. (e) OB-LNN [8] uses rotated hyper-boxes. (f) L_1, L_∞-SLLP [9] uses polytopes. (g) MP/CL [10] and DMNN [12] present complex non-linear surfaces as decision boundaries due to dendrite processing and argmax operator as activation function. In general, morphological neural networks have superior classification ability to classical single-layer perceptrons, and some morphological networks such as (c) and (g) have at least the same capability as multilayer perceptrons. Even so, morphological neurons are less popular than classical perceptrons among practitioners, engineers and researchers. We hope this work can help them to see the benefits of morphological neurons, particularly dendrite morphological neurons (g). Note that all acronyms are defined in the references.

III. DMN trained by SGD

This section presents our proposal. First, we describe the neural architecture of a conventional DMN. This architecture is modified to be trained by SGD [18]. The SGD training method is presented in some detail for practitioners who are interested in using this method. The code and a demo of the training algorithm are online at [3].

A. Neural Architecture

1) Conventional DMN: A DMN has been shown to be useful for pattern classification [11], [12], [10]. This neuron works similarly to a radial basis function network (RBFN) but uses hyper-boxes instead of radial basis functions and has only one layer. Each dendrite generates a hyper-box and the dendrite that is most active is the one whose case is closer to the input pattern. The output y of this neuron (see Fig. 3) is a scalar given by

$$y = \arg\max_{c} \left(d_{n,c} \right), \tag{1}$$

where n is the dendrite number, c is the class number and $d_{n,c}$ is the scalar output of a dendrite given by

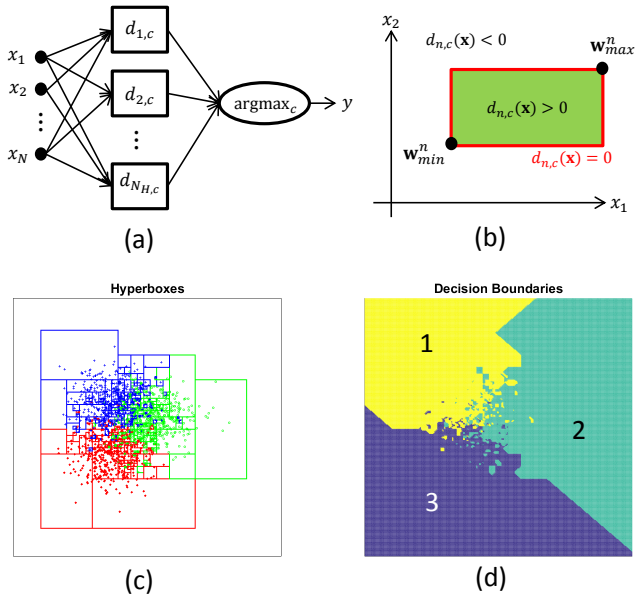

Fig. 3. We show: (a) the architecture of a conventional DMN; (b) an example of a hyper-box in 2D generated by its dendrite weights. A dendrite output is positive when the pattern is in the green region (inside the corresponding hyper-box), it is zero when the pattern is in the red region (within the hyper-box boundary), and it is negative in the white region (outside the hyper-box); (c) an example of DMNs hyper-boxes for a classification problem; (d) the decision boundaries generated by a DMN for the same classification problem in (c). Note that decision boundaries can be non-linear and a class region can consist of disconnected regions. These properties allow a DMN to solve any classification problem separating classes by a given tolerance [15].

$$d_{n,c} = \min\left(\min\left(\mathbf{x} - \mathbf{w}^n_{min}, \mathbf{w}^n_{max} - \mathbf{x}\right)\right), \tag{2}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}_{min}$ and $\mathbf{w}_{max}$ are dendrite weight vectors. The min operators together check if $\mathbf{x}$ inside the hyper-box is limited by $\mathbf{w}_{min}$ and $\mathbf{w}_{max}$ as the extreme points, according to Fig. 3. If $d_{n,c} > 0$, $\mathbf{x}$ is inside the hyper-box; if $d_{n,c} = 0$, $\mathbf{x}$ is somewhere in the hyper-box boundary; otherwise, it is outside. The inner $\min$ operator in (2) compares elements of the same dimension between the two vectors, generating another vector; while the outer $\min$ operator takes the minimum among all dimensions, generating a scalar. The activation function is given by the$\arg\max$ operator proposed by [10] instead of the hard limit function that is commonly used in SLMP [7]. This argmax operator allows the building of more complex decision boundaries than simple hyper-boxes (see Fig. 3). It is worth mentioning that if (1) produces several maximums, the argmax operator takes the first maximum as as index class to which the input pattern is assigned. Qualitatively speaking, this occurs when $\mathbf{x}$ is equidistant to more than one hyper-box.

2) DMN with Softmax Layer: The principal problem in training a conventional DMN by SGD is in calculating the gradients when we have an argmax function in the output because argmax is a discrete function. This can be easily overcome by using a softmax layer instead of the argmax function. The softmax layer normalises the dendrite outputs such that these outputs are restricted between zero to one and can be interpreted as a measurement of likelihood Pr so that a pattern $\mathbf{x}$ belongs to the class c, as follows:

$$Pr_c\left(\mathbf{x}\right) = \frac{\exp\left(d_c(\mathbf{x})\right)}{\sum_{k=1}^{N_C} \exp\left(d_k(\mathbf{x})\right)}, \tag{3}$$

The assigned class is taken based on these probabilities by

$$y = \arg\max_c \left(Pr_c\left(\mathbf{x}\right)\right), \tag{4}$$

We propose making dendrite clusters d_c in order to have several hyper-boxes per class, as in [17], [19]. In Fig. 4, we show our proposed neural architecture. If a particular classification problem has N_c classes, then we would need N_c dendrite clusters, one per class. Each dendrite cluster output takes the maximum among their dendrites h, as follows

$$d_c(\mathbf{x}) = \max_k \left(h_{k,c}(\mathbf{x})\right), \tag{5}$$

where a dendrite k of a cluster c is given by

$$h_{k,c}(\mathbf{x}) = \min\left(\min\left(\mathbf{x} - \mathbf{w}_{k,c}, \mathbf{w}_{k,c} + \mathbf{b}_{k,c} - \mathbf{x}\right)\right), \tag{6}$$

Note that we change the way to codify a hyper-box mathematically. In conventional DMN, a hyper-box is represented by its extreme points $\mathbf{w}_{min}$ and $\mathbf{w}_{max}$. However, here a hyper-box is represented by its lowest extreme point $\mathbf{w}_{k,c}$ and its vector $\mathbf{b}_{k,c}$ which determines the hyper-box size for each dimension.

B. Training Method

A key issue of morphological neural networks is their training. We need to determine automatically the number of hyper-boxes for each dendrite cluster, and the dendrite parameters $\mathbf{w}_{k,c}$ and $\mathbf{b}_{k,c}$. This is a great difference with respect to classical perceptrons where only the learning parameters are determined during training. Several training approaches for morphological neurons have been proposed, as shown in section 2. This paper proposes to initialise the hyper-boxes based on heuristics and optimise the dendrite parameters by SGD [18]. Section 4 shows that this method is better than training based only on heuristics.

We first explain the optimisation stage. The objective function is the cost J resulted by the softmax layer, as follows

$$J = -\sum_{q=1}^{Q}\sum_{c=1}^{N_C} 1\left\{t_q = c\right\} \log\left(Pr_c\left(\mathbf{x}_q\right)\right), \tag{7}$$

where Q is the total number of training samples, q is the index for a training sample, t_q is the target class for that training sample, and $1\left\{\cdot\right\}$ is the indicator function so that $1\left\{true\ statement\right\} = 1$, and $1\left\{false\ statement\right\} = 0$. The SGD method minimises this cost (7) by the gradients $\frac{dJ}{d\mathbf{w}_{k,c}}$ and $\frac{dJ}{d\mathbf{b}_{k,c}}$, which are given by

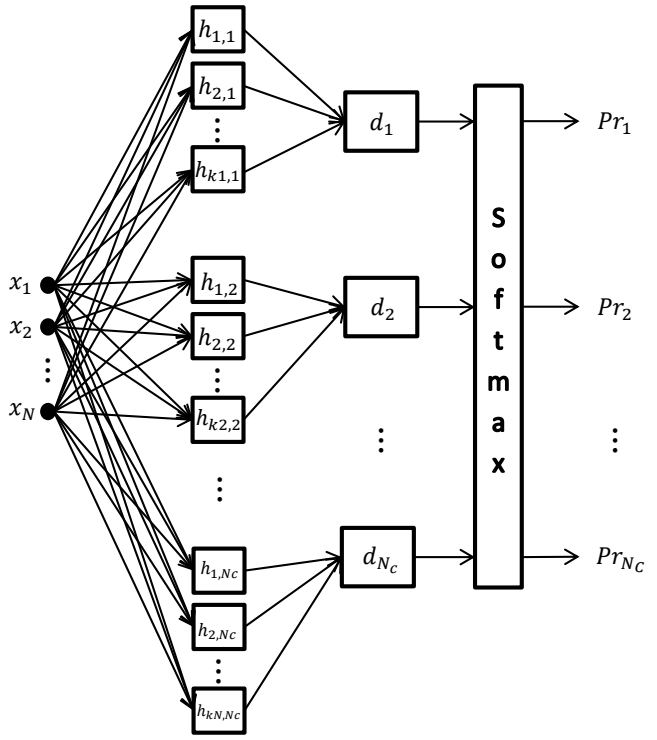

Fig. 4. Neural architecture for a DMN with a softmax layer. Each class corresponds to one dendrite cluster d_c.

$$\frac{dJ}{d\mathbf{w}_{k,c}} = -\sum_{q=1}^{Q_{batch}} \left[\begin{array}{ccccccc} 0 & \dots & 0 & f_{w_{k,c}} & 0 & \dots & 0 \end{array}\right]^T, \quad (8)$$

$$\frac{dJ}{d\mathbf{b}_{k,c}} = -\sum_{q=1}^{Q_{batch}} \left[\begin{array}{ccccccc} 0 & \dots & 0 & f_{b_{k,c}} & 0 & \dots & 0 \end{array}\right]^T, \quad (9)$$

where

$$f_{w_{k,c}} = \begin{cases} 0 & k \neq k* \\ -\left(1\left\{t_q = c\right\} - Pr_c\left(\mathbf{x}_q\right)\right) & k = k* \wedge j* = 1 \\ 1\left\{t_q = c\right\} - Pr_c\left(\mathbf{x}_q\right) & k = k* \wedge j* = 2 \end{cases}, \quad (10)$$

$$f_{b_{k,c}} = \begin{cases} 0 & k \neq k* \vee j* = 1 \\ 1\left\{t_q = c\right\} - Pr_c\left(\mathbf{x}_q\right) & k = k* \wedge j* = 2 \end{cases}, \quad (11)$$

where $k*$ is the dendrite index that generates the maximum response in the dendrite cluster c, $j*$ is the column in the matrix $[\mathbf{x} - \mathbf{w}_{k,c}, \mathbf{w}_{k,c} + \mathbf{b}_{k,c} - \mathbf{x}]$ showing the minimum and the position of $f_{w_{k,c}}$ and $f_{b_{k,c}}$ in the vectors (8) and (9) is determined by the row number in the matrix $[\mathbf{x} - \mathbf{w}_{k,c}, \mathbf{w}_{k,c} + \mathbf{b}_{k,c} - \mathbf{x}]$ showing the minimum. Due to the min and max operators in DMN, most gradient elements are zero. This means that only one parameter of $\mathbf{w}_{k,c}$ and $\mathbf{b}_{k,c}$ and one dendrite in each cluster is updated by the gradient descent method when the cost is calculated by only one training sample. However, in practice, we prefer to apply the gradient descent in batches with Q_{batch} training samples taken randomly.

We explain the different initialisation methods for dendrite parameters. Unlike classical MLP where learning parameters are initialised randomly, here DMN allows us to propose different initialisation methods because dendrite parameters have a clear geometrical interpretation: they are hyper-boxes. Hence, we can reuse training methods based on heuristics to give a first approximation at the beginning of the gradient descent iterations. This paper proposes the following four initialisation methods, showing an illustration of these methods in Fig. 5.

- **HpC initialisation** [10]. Each training set for the same class is enclosed by a hyper-box (Hyper-box per Class) with a distance margin $M \in (-0.5, \infty)$. This is the most simple way to initialise dendrites. The drawback is that it restricts the maximum number of dendrites to be equal to the number of classes, and sometimes the pattern distribution may require a greater number of dendrites. Therefore, we propose the following methods that generate more hyper-boxes per class to capture the data structure.
- **dHpC initialisation**. This method divides each hyper-box which was previously generated by the above method. The number of hyper-boxes produced by this method is equal to $2^{n_d}N_c$, where n_d is the number of dimensions that are divided. For example, if $n_d = 1$, the algorithm divides the hyper-boxes in half with respect to the first dimension, producing $2N_c$ hyper-boxes. If $n_d = 2$, it divides the hyper-boxes in half with respect to the first and second dimensions, producing $4N_c$ hyper-boxes, and so on.
- **D&C initialisation** [12]. The dendrite parameters are set equal to the dendrites produced by the Divide and Conquer training method proposed in [12]. This algorithm encloses all patterns in one hyper-box with a distance margin of $M \in [0, \infty)$ and divides this hyper-box into 2^N sub-hyper-boxes if there is misclassification. This last procedure is called recursively for each sub-hyper-box until each pattern is classified with a given tolerance error E_0. This method is computationally more expensive but it captures the data distribution in a better way. This paper deploys an improved version of this algorithm which can avoid the curse of dimensionality and overfitting. This new algorithm will be published later (Guevara & Sossa, 2016, Personal Communication).
- **K-means initialisation**. The classical k-means algorithm is applied to obtain S clusters for each training set of the same class. The k-means clusters are transformed to hyper-boxes, and we use these to set dendrite parameters (we do the inverse process proposed in [19]). We use

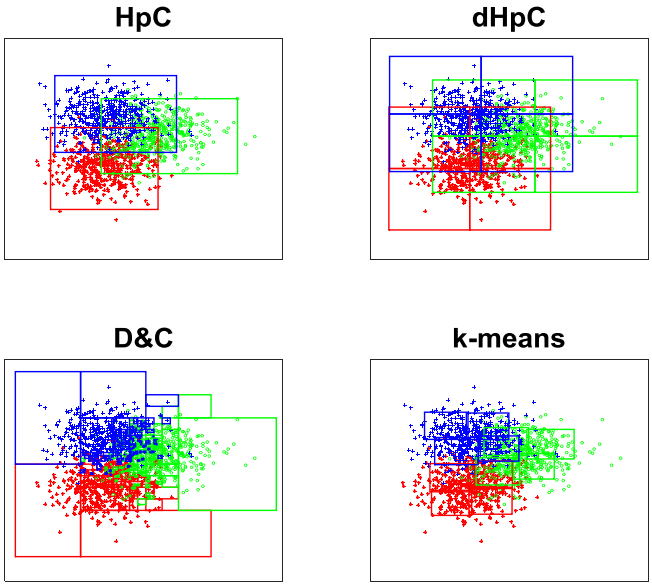

Fig. 5. We illustrate the four initialisation methods. The HpC method initialises dendrite parameters, enclosing each class with a hyper-box with a margin (even negative). dHpC is the same as the previous method but allows division of hyper-boxes in half with respect to some dimensions, generating more hyper-boxes per class. The D&C method is specifically to use D&C training to initialise dendrites. The k-means method first makes a clustering to define where to put the initial hyper-boxes using the classical k-means method.

Gaussians as radial basis functions for clustering:

$$\phi_{k,c}(\mathbf{x}) = \exp(-\beta_{k,c} \left\| \mathbf{x} - \mu_{k,c} \right\|^2), \tag{12}$$

The cluster centre $\mu_{k,c}$ and width $\beta_{k,c}$ are used to calculate the dendrite parameters to form a hyper-box for each Gaussian cluster, as follows:

$$\mathbf{w}_{k,c} = \mu_{k,c} - \mathbf{\Delta}_{k,c}, \tag{13}$$

$$\mathbf{b}_{k,c} = 2\Delta_{k,c}, \tag{14}$$

$$\mathbf{\Delta}_{k,c} = \sqrt{\frac{-\log(y_0)}{\beta_{k,c}}}, \tag{15}$$

where $y_0 \in (0, 1]$ determines the hyper-box size (note that vector operations in (15) are element-wise). This initialisation method also increases computational cost but can create more complex learning models than HpC and dHpC methods.

IV. Experiments

In this section, we compare our proposal with other training methods for DMN and popular machine learning methods based on synthetic and real datasets for classification. The results demonstrate that our proposal obtains systemically better accuracy than other training methods for DMNs and achieves a similar performance to popular machine learning methods. Furthermore, we discuss the advantages and disadvantages of each hyper-box initialisation method for our proposal to determine which is the best. The reader can reproduce the experimental results of this paper by using the online code provided at [3]. At the end of the section, we present special experiments to show the best asset of the DMNs regarding MLPs.

A. Synthetic and Real Datasets

Our proposed method is first applied to synthetic datasets with high overlapping. We have chosen datasets where overlapping is present because overfitting is more relevant in this kind of problem. The first synthetic dataset is called A and is generated by two Gaussian distributions with standard deviation equal to 0.9. The first class is centred around (0,0); the second class is centred around (1,1). The second synthetic dataset is called B and consists of three classes which follow a Gaussian distribution with standard deviation equal to 1.0. The class centres are: (-1,-1), (1,1) and (-1,2).

We also use some real datasets to test the performance of our proposal. Most of these datasets were taken from the UCI Machine Learning Repository [20]. We also use a subset of MNIST [21] and CIFAR10 [22] datasets for tasks with high dimensionality. We describe them briefly. The Iris dataset is a classical non-linearly separable problem with three types of iris plant and four describing features. The Liver dataset consists of detecting liver disorder (two classes) using six features that relate to blood tests and alcoholic drinking. The Glass dataset has the purpose of identifying the glass type in crime scenes using 10 properties of glass. The number of classes is six. The Page Blocks dataset needs to classify blocks of documents into five categories based on 10 block geometrical features. The Letter Recognition dataset has 26 letters from different fonts which must be identified by 16 geometrical and statistical features. The Mice Protein expression dataset classifies mice into eight classes based on features such as genotype, behaviour and treatment. In this work, we only use the expression levels of 77 proteins as features for our experiments. We apply the proposed method to two image classification tasks taking pixels as features (without intermediate feature extraction). The MNIST dataset is a classical recognition task for handwritten digit recognition (0 to 9) with an image size of 28×28 pixels. The CIFAR10 requires recognition of 10 animals and vehicles by means of colour images of 32×32 pixels. It is worth mentioning that in some of these datasets we found duplicates; we deleted them for the experiments. Table I summarises the dataset properties in terms of number of features N, number of classes N_c, number of training samples Q_{train} and number of testing samples Q_{test}.

B. Comparison with Other DMN Training Methods

This subsection analyses the performance of DMN trained by SGD in comparison with other training methods based just on heuristics. We also compare the performance of the different ways to initialise hyper-boxes that we proposed in section 3. These comparisons are made based on the datasets described above.

TABLE I
WE SUMMARISE THE DATASET PROPERTIES.

Dataset	N	N_c	Q_{train}	Q_{test}
A	2	2	1000	200
B	2	3	1500	300
Iris	4	3	120	30
Liver	6	2	300	45
Glass	10	6	170	44
Page Blocks	10	5	4450	959
Letter Recog.	16	26	15071	3917
Mice Protein	77	8	900	180
MNIST	784	10	1000	200
CIFAR10	3072	10	1000	200

We measure performance in terms of the training accuracy A_{train}, the testing accuracy A_{test}, training time t_{train} and the model complexity N_H/Q_{train}. This last quantity is defined as the ratio of the number of hyper-boxes N_H by the number of training samples Q_{train}. If this ratio is equal to 1 then the neuron uses a hyper-box for each training sample. In this extreme case, the model is very complex. In contrast, if this ratio is close to zero, it means that the number of hyper-boxes is much smaller than the number of training samples. Therefore, the model is simple. Ideally, we are looking for a morphological neuron with relatively few hyper-boxes ($N_H/Q_{train} \simeq 0$), with the greatest testing accuracy A_{test} and with $A_{train} \sim A_{test}$; that is, a simple and well generalised learning model.

We compare our proposal with the principal algorithms: Elimination (SMLP) [15], Hyper-box per Class (HpC) (a simplified version of the algorithm proposed in [10]), and Divide and Conquer (D&C) [12]. Elimination training encloses all patterns in a hyper-box and eliminates misclassified patterns by enclosing them with several other hyper-boxes. HpC training encloses the patterns of each class by using a hyper-box. D&C training is based on a Divide and Conquer strategy; this training begins enclosing all patterns in one hyper-box, then divides it into 2^N hyper-boxes if there is misclassification. This last procedure is called recursively for each sub-hyper-box until patterns are classified with a given tolerance error. It is important to note that all these training methods are based on heuristics, not on optimisation methods, so we could expect that SGD training performs better. Additionally, we compare the four initialisation methods proposed in section 3 for SGD training, which are HpC, dHpC, D&C and k-means.

Fig. 6 summarises our results, showing that SGD training systemically obtains greater testing accuracies for each dataset than the other heuristics. Furthermore, it is seen that when initialising SGD training with the dHpC method, we obtain better testing accuracies on average than with other initialisation methods. On average, the best initialisation method in terms of the least number of dendrites is the HpC method and in terms of least training time it is the k-means method. We can conclude that SGD presents better results than solely heuristic-based methods to train DMNs.

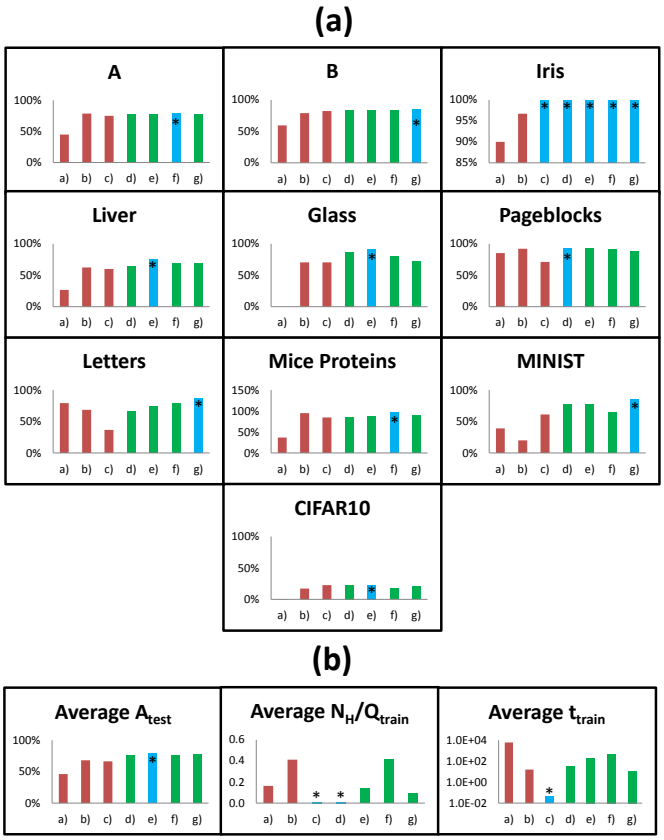

Fig. 6. Our experimental results are summarised. (a) We show the testing accuracies for each dataset obtained by the training methods based on heuristics: a) SMLP, b) D&C and c) HpC; and the SGD training is executed from four different initialisation methods: d) SGD (HpC), e) SGD (dHpC), f) SGD (D&C) and g) SGD (k-means). (b) We show the average among all datasets for testing accuracy A_{test}, model complexity N_H/Q_{train} and training time t_{train}. We mark the best value with blue (with asterisk). Note that quantities are sometimes close to zero, meaning that bars disappear.

C. Comparison with Other Machine Learning Methods

This subsection compares the performance of popular machine learning methods with our proposal: DMN-SGD. These comparisons are made based on the datasets described above. As popular machine learning methods, we chose two-multilayer perceptrons (MLP) [23], support vector machine (SVM) [24] and radial basis function network (RBFN) [25]. We summarise our results in Fig. 7, showing that DMN-SGD obtains the greatest testing accuracies for some datasets (A, B, Iris and Liver). For other datasets (Glass, Pageblocks, Letters and Mice Protein), it presents accuracies close to the best result; and for the remaining datasets (MNIST and CIFAR10), it obtains the worst accuracy. This infers that the DMN-SGD algorithm works better for low dimensional datasets (fewer features) than for high dimensional datasets. On the other hand, on average, the testing accuracy for DMN-SGD is the second best behind the MLP model for these 10 datasets. The main drawback is that DMN-SGD needs more model parameters N_{pm} than the other machine learning models, but DMN-SGD has better training time than MLP on average. This

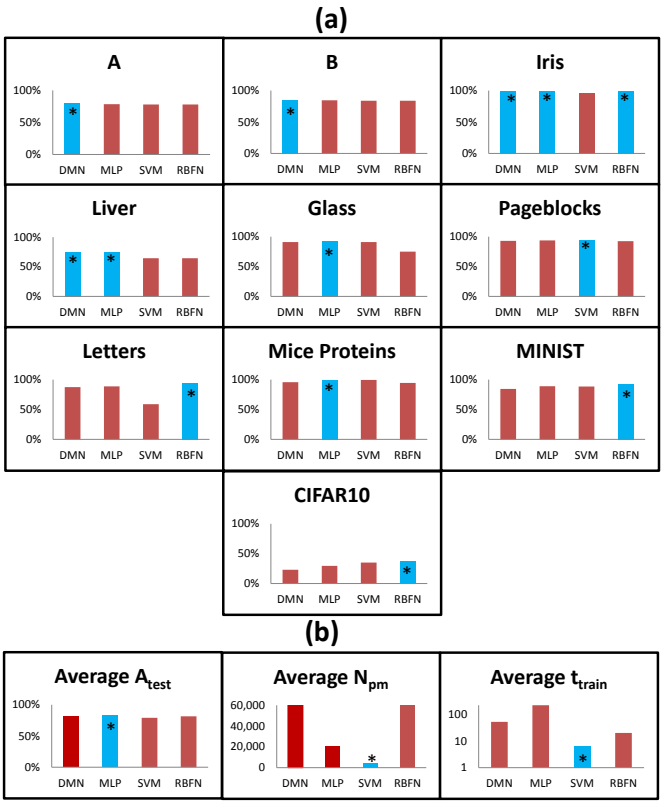

Fig. 7. We compare our proposal DMN-SGD with popular machine learning methods, summarising the experimental results. (a) We show the testing accuracies for each dataset obtained by the following machine learning methods: DMN, MLP, SVM and RBFN. (b) We show the average among all datasets for testing accuracy A_{test}, number of learning parameters N_{pm} and training time t_{train}. We mark the best value with blue (with asterisk). Note that quantities are sometimes close to zero, whereby bars disappear.

concludes that DMN trained by SGD can be another useful technique in the machine learning toolbox for practitioners, because it obtains competitive results with respect to popular machine learning methods for classification problems.

D. DMN vs MLP

The previous subsection has shown that the DMN performance is competitive with popular machine learning techniques. In particular, MLP has marginally better classification than DMN on average. There are some problems where DMN far outstrips MLP. The purpose of this subsection is to show empirically the greatest benefit of the DMNs.

According to our proposal, training both DMN and MLP requires two steps: 1) initialisation of learning parameters in the model and 2) optimising them. In the MLP, initialisation is a very simple step: choose the initial parameters randomly following some convenient probability distribution [26], [27]; rarely do these random initial values give a good decision boundary. Therefore, optimisation should search more to find the best parameters in MLP. Instead, in the DMN, initialisation of dendritic weights exploits the geometric interpretation of the weights (as hyper-boxes) to facilitate optimisation, solving

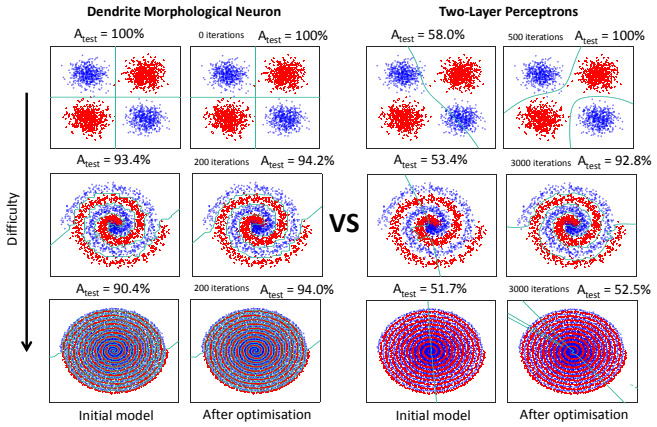

Fig. 8. This figure illustrates why the dendritic weights initialisation is the greatest benefit of the DMN. This initialisation: 1) makes the optimisation step easier, reducing the number of required iterations during the optimisation step; 2) solves problems which classical perceptrons cannot, or at least, it takes them much more work. These properties are due to geometric interpretation of the dendritic weights as hyper-boxes that allow intialisation methods based on heuristics. This cannot happen with MLP.

problems that classical perceptrons cannot, or at least, at reduced work costs. This is the greatest benefit of DMNs. Here is where heuristic methods excel the parameter initialisation.

To illustrate this, we propose to evaluate the performance of both models (MLP and DMN) using three synthetic classification problems: a type XOR problem, a double spiral with two laps and a double spiral with 10 laps. In particular, the spirals are difficult problems for MLPs because they tend to create open decision boundaries with no laps.

The experimental results are shown in Fig. 8. It is clearly noticeable that in all three cases, the optimisation requires more iterations in MLPs w.r.t DMNs. We also noticed that optimisation in DMN is just a process for refining the initial model, while optimisation in MLP is a more decisive step because initial models are almost always misclassified. Finally, the most important observation is that an MLP cannot solve the double spiral with 10 laps (or more than 10), even though the decision boundary follows a simple mathematical pattern. Instead, DMN initialises the decision boundary with a good approximation by a D&C strategy, allowing the optimisation step to refine the model after only a few iterations. Of course, more research is needed to see in which real problems this would happen. Here, we just pave the way towards this research.

V. Conclusions

The main contribution of this paper is that the neural architecture of a dendrite morphological neuron (DMN) was extended to be trained by stochastic gradient descent (SGD) for classification problems. This was made possible by adding a softmax layer. We proposed and evaluated four different methods to initialise dendrite parameters at the beginning of training by SGD. Some of these initialisation methods are based on heuristic-based training methods which exploit the

geometrical interpretation of dendritic weights as hyper-boxes. We performed several experiments with standard datasets for machine learning to compare our approach with previous training methods for DMN, and with popular machine learning algorithms, especially with MLP. The experiments show that our approach trained by SGD systematically obtains better classification accuracies than the training methods based solely on heuristics. We show that these heuristics can be useful to initialise dendrite parameters before optimising them. This is an advantage over MLP where learning parameters are only initialised randomly. In fact, we show that DMN can solve some problems that MLP cannot, or at least, it involves a lot more work. Furthermore, our results show that these DMNs trained by SGD can compete with popular machine learning algorithms like SVM, MLP and RBFN. On average, DMNs present a better performance than SVM, a similar performance to RBFN and are inferior to MLP. The advantage is that DMNs spend lower computational cost because of min and max operators instead of multiplications. This simplifies the DMN implementation on micro-controllers and FPGAs. We think more exploration must be done to improve results in terms of new neural architectures and optimisation-based training in DMNs.

ACKNOWLEDGMENT

The authors would like to acknowledge the support for this research work by the UPIITA and the CIC, both schools belonging to the Instituto Politécnico Nacional (IPN). This work was economically supported by SIP-IPN [grants numbers 20160945 and 20161116]; and CONACYT [grant numbers 155014 (Basic Research) and 65 (Frontiers of Science)].

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[3] E. Zamora, "Dendrite Morphological Neurons Trained by Stochastic Gradient Descend - Code," https://github.com/ezamorag, 2016.

[4] 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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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

[11] H. Sossa and E. Guevara, *Pattern Recognition: 5th Mexican Conference, MCPR 2013, Querétaro, Mexico, June 26-29, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Modified Dendrite Morphological Neural Network Applied to 3D Object Recognition, pp. 314–324.

[12] ——, "Efficient training for dendrite morphological neural networks," *Neurocomputing*, vol. 131, pp. 132 – 142, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213010916

[13] 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

[14] 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.

[15] 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.

[16] D. Chyzhyk and M. Graña, *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. Optimal Hyperbox Shrinking in Dendritic Computing Applied to Alzheimer's Disease Detection in MRI, pp. 543–550.

[17] 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

[18] G. Montavon, G. B. Orr, and K.-R. Müller, Eds., *Neural Networks: Tricks of the Trade, Reloaded*, 2nd ed., ser. Lecture Notes in Computer Science (LNCS). Springer, 2012, vol. 7700.

[19] H. Sossa, G. Cortés, and E. Guevara, *New Radial Basis Function Neural Network Architecture for Pattern Classification: First Results*. Cham: Springer International Publishing, 2014, pp. 706–713.

[20] A. Asuncion and D. Newman, "UCI Machine Learning Repository. Irvine, CA: University of California, Department of Information and Computer Science," http://www.ics.uci.edu/ mlearn/MLRepository.html, 2007.

[21] C. LeCun, Y. Corinna and C. Burges, " The MNIST dataset of handwritten digits. NewYork: NYU, Google Labs and Microsoft Research," http://yann.lecun.com/exdb/mnist/, 1998.

[22] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images. MIT and NYU," https://www.cs.toronto.edu/ kriz/cifar.html, 2009.

[23] 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. [Online]. Available: http://dl.acm.org/citation.cfm?id=104279.104293

[24] 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

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

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-10)*, Y. W. Teh and D. M. Titterington, Eds., vol. 9, 2010, pp. 249–256. [Online]. Available: http://www.jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.