Towards Dendrite Spherical Neurons for Pattern Classification

Wilfrido Gómez-Flores[1(✉)] and Juan Humberto Sossa-Azuela[2,3]

[1] Centro de Investigación y de Estudios Avanzados del IPN, Unidad Tamaulipas, 87130 Ciudad Victoria, Tamaulipas, Mexico
wgomez@cinvestav.mx

[2] Instituto Politécnico Nacional, Centro de Investigación en Computación, 07738 Mexico City, Mexico
hsossa@cic.ipn.mx

[3] Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias, Av. General Ramón Corona 2514, Zapopan, Jalisco, Mexico

Abstract. This paper introduces the Dendrite Spherical Neuron (DSN) as an alternative to the Dendrite Ellipsoidal Neuron (DEN), in which hyperspheres group the patterns from different classes instead of hyperellipses. The reasoning behind DSN is simplifying the computation of DEN architecture, where a centroid and covariance matrix are two dendritic parameters, whereas, in DSN, the covariance matrix is replaced by a radius. This modification is useful to avoid singular covariance matrices since DEN requires measuring the Mahalanobis distance to classify patterns. The DSN training consists of determining the centroids of dendrites with the k-means algorithm, followed by calculating the radius of dendrites as the mean distance to the two nearest centroids, and finally determining the weights of a softmax function, with Stochastic Gradient Descent, at the output of the neuron. Besides, the Simulated Annealing automatically determines the number of dendrites that maximizes the classification accuracy. The DSN is applied to synthetic and real-world datasets. The experimental results reveal that DSN is competitive with Multilayer Perceptron (MLP) networks, with less complex architectures. Also, DSN tends to outperform the Dendrite Morphological Neuron (DMN), which uses hyperboxes. These findings suggest that the DSN is a potential alternative to MLP and DMN for pattern classification tasks.

Keywords: Dendrite Morphological Neuron · Spherical dendrite · Simulated Annealing · Pattern classification

1 Introduction

Artificial Neural Networks (ANN) are mathematical models inspired by the biological neurons in the nervous system of the animals, which can be described as mapping an input space to an output space [7]. Probably, the Multilayer Perceptron (MLP) is the most common ANN used in practice for pattern classification tasks. MLP training requires adjusting the synaptic weights of each

K. M. Figueroa Mora et al. (Eds.): MCPR 2020, LNCS 12088, pp. 14–24, 2020.
https://doi.org/10.1007/978-3-030-49076-8_2

neuron by minimizing a loss function (e.g., cross-entropy), where the backpropagation algorithm is often used. The inner product between the neuron inputs and the synaptic weights produces a linear combination that is modified by a nonlinear activation function (e.g., the sigmoid function). Thus, the MLP divides the input space with a hypersurface, which is built by combining the responses of several neurons distributed in one or more hidden layers.

In nonlinear separability scenarios, the MLP could require a complex architecture to separate the input space accurately. The Dendrite Morphological Neuron (DMN) is an alternative technique that reduces the complexity of the classification models since nonlinear classification problems can be solved by using a single neuron. The morphological processing involves minimum and maximum operations, which can generate complex nonlinear decision boundaries [8].

A typical DMN has dendrites defined as hyperboxes in $\mathbb{R}^D$, where D is the dimensionality of the input space. A set of hyperboxes can model each class pattern, where the minimum and maximum operations determine if an input pattern is inside of a hyperbox; therefore, the input pattern is assigned to the class of the most active dendrite. The DMN training consists of distributing the hyperboxes over the input space such that every class pattern is covered accurately, where heuristic methods [8], evolutionary computation [3], and stochastic gradient descent (SGD) [9] have been used for this purpose.

Because DMN uses hyperboxes, the produced decision boundaries are complex piecewise linear functions. In order to obtain smoother decision boundaries, it is feasible to replace the hyperboxes with other geometrical shapes. In this context, Arce *et al.* [2] proposed a neuronal model called Dendrite Ellipsoidal Neuron (DEN), where an input pattern is assigned to the class of the dendrite (i.e., hyperellipse) with the minimum Mahalanobis distance. A hyperellipse is defined by two parameters: centroid and covariance matrix. In DEN, the centroid positions within the input space are defined by the k-means algorithm, in which k is the number of dendrites within a class. Next, for obtaining rotated hyperellipses, the covariance matrix of each cluster is calculated. Note that a class is modeled by k dendrites, where a dendrite clusters only a fraction of samples from the class. Therefore, as the value of k increases, the number of samples in the dendrite decreases, so that there could be variables with zero variance, generating a singular covariance matrix. Consequently, the calculation of the Mahalanobis distance cannot be performed for that dendrite since it is required to invert its covariance matrix.

To overcome this inconvenience, we propose a simplification of the DEN model by using spheres instead of ellipses to get a new neuronal model called Dendrite Spherical Neuron (DSN), in which the full covariance matrix of a dendrite is replaced by a radius that depends on the closeness among centroids. Moreover, the computation of the DNS response is more straightforward than DEN because covariances and matrix inversions are no longer necessary.

2 DSN Architecture

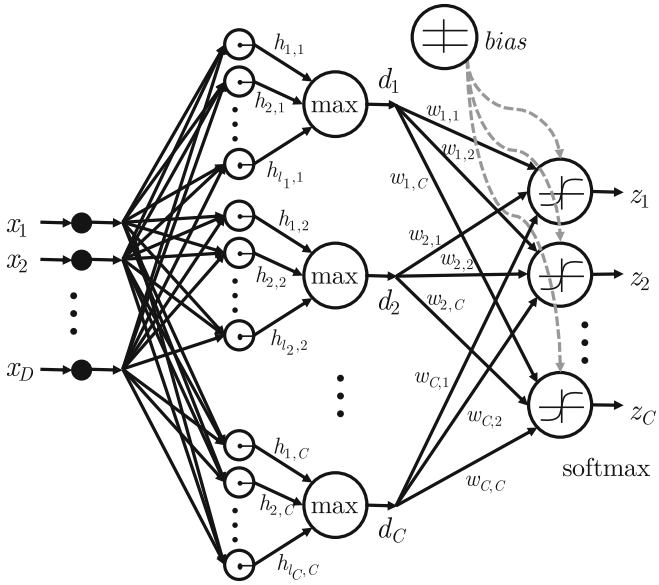

Fig. 1. Neural architecture for a DSN with softmax function at the output. The jth class is modeled by the dendrite cluster with response d_j, for $j = 1, \ldots, C$ classes.

Fig. 1 shows the neural architecture for a DSN, in which each class is represented by a cluster of dendrites, that is, a set of hyperspheres in $\mathbb{R}^D$. The DSN output is performed the linear combination of dendrite responses d_j, for $j = 1, \ldots, C$ classes, and the softmax function gives the probability of the input pattern $\mathbf{x} = [x_1, \ldots, x_D]^T$ belongs to the jth class. Thus, the assigned class is given by the maximum probability rule [6]:

$$\hat{t}_j = \underset{j=1,\ldots,C}{\arg\max} \left(z_j(\mathbf{x}) \right), \tag{1}$$

where z_j is the response of the jth output node defined as

$$z_j(\mathbf{x}) = \sigma \left(w_{0j} + \sum_{k=1}^{C} w_{k,j} d_j(\mathbf{x}) \right), \quad j = 1, \ldots, C, \tag{2}$$

where $\sigma(\cdot)$ is the softmax function, $w_{k,j}$ is a weight value to connect the kth cluster to the jth output node, w_{0j} is the bias, and d_j is the output of the jth dendrite cluster:

$$d_j(\mathbf{x}) = \max_{i=1,\ldots,l_j} \left(h_{i,j}(\mathbf{x}) \right), \tag{3}$$

where $h_{i,j}$ is the output of the ith dendrite for the jth class:

$$h_{i,j}(\mathbf{x}) = r_{i,j} - \left\| \mathbf{x} - \mathbf{c}_{i,j} \right\|^2, \tag{4}$$

where $\|\cdot\|$ is the Euclidean norm, $\mathbf{c}_{i,j} \in \mathbb{R}^D$ is the centroid of the dendrite, and $r_{i,j} > 0$ is its corresponding radius.

Figure 2 illustrates the three possible responses of a dendrite given in Eq. 4. A dendrite obtains its maximum response when $\mathbf{x} = \mathbf{c}$, that is, $h(\mathbf{x}) = r$. As $\mathbf{x}$ moves away from the centroid, the dendrite response decreases to zero on its boundary, and becomes negative outside of the dendrite region. Thus, the most active dendrite cluster can be identified with Eq. 3.

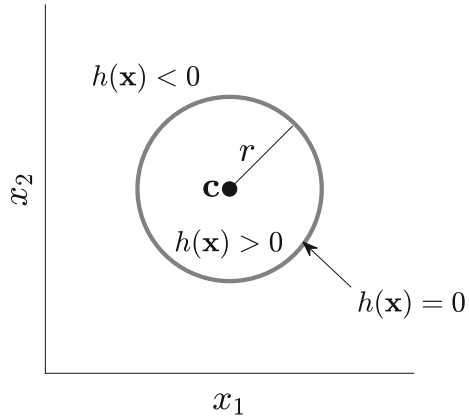

Fig. 2. A hypersphere in 2D generated by its dendrite parameters $\mathbf{c}$ and r. The response is positive when the pattern $\mathbf{x}$ is inside of the hypersphere, it is zero when $\mathbf{x}$ is on the hypershpere boundary, and it is negative when $\mathbf{x}$ is outside of the hypersphere.

3 DSN Training

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a training set with N observations, where the ith sample is a D-dimensional vector $\mathbf{x}_i = [x_{i,1}, \ldots, x_{i,D}]^T$, which is associated to a class label $t_i \in \{1, \ldots, C\}$.

Algorithm 1 shows the pseudocode for training a DSN based on the k-means algorithm and SDG. First, the centroids of dendrites are calculated with the k-means algorithm (lines 3–8), where the parameter $k = l_j$ is the number of hyperspheres in a class. Next, to reduce the overlap between dendrite regions, the mean distance to the two nearest centroids determines the radius of a dendrite (lines 9–10). Finally, the cluster dendrite responses are calculated from the entire training set (Eqs. 3 and 4), which are used to obtain the weights of the softmax function by minimizing the cross-entropy loss function with SDG (lines 12–13).

Notice that Algorithm 1 requires the number of dendrites per class, which is problem-dependent and is typically not known *a priori*. It is desirable a DSN configuration with a reduced number of dendrites and a high classification rate. This goal can be achieved by using an optimization procedure to maximize the

Algorithm 1: DSN training based on k-means and SDG.

Input: Training patterns $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$; targets $\mathbf{t} = [t_1, \ldots, t_N]$; number of classes C; number of dendrites per class $[l_1, \ldots, l_C]$
Output: A structure DSN

1 DSN $\leftarrow \emptyset$ `// initialize dendrite structure`
2 $\mathbf{C} \leftarrow \emptyset$ `// initialize temporal array of centroids`
3 **for** $j = 1, \ldots, C$ **do**
4 Get patterns of the jth class from $\mathbf{X}$ to obtain the subset $\mathbf{X}_j$
5 Obtain l_j centroids with k-means from $\mathbf{X}_j$ to get $C_j = [\mathbf{c}_{1,j}, \ldots, \mathbf{c}_{l_j,j}]$
6 Concatenate centroids $\mathbf{C} \leftarrow [\mathbf{C} + C_j]$
7 Save centroids: DSN.$\mathbf{c}_{i,j} \leftarrow \mathbf{c}_{i,j}$, $\forall i = 1, \ldots, l_j$
8 **end for**
9 Measure the pairwise distance between centroids in $\mathbf{C}$
10 Calculate the mean distance of $\mathbf{c}_{i,j}$ to the 2-nearest centroids to get its corresponding radius $r_{i,j}$, $\forall i, j$
11 Save radii: DSN.$r_{i,j} \leftarrow r_{i,j}$, $\forall i, j$
12 Obtain the cluster dendrite responses $d_j(\mathbf{x})$, $j = 1, \ldots, C$, for all samples in $\mathbf{X}$ to obtain $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$
13 Calculate the softmax weights $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_C]$ with SDG and cross-entropy from the tuple $(\mathbf{Y}, \mathbf{t})$
14 Save weights: DSN.$\mathbf{W} \leftarrow \mathbf{W}$
15 **return** DSN

classification accuracy with few dendrites. Herein, the Simulated Annealing (SA) algorithm is employed to tune the DSN configuration automatically.

SA is a stochastic local search method for global combinatorial optimization, which allows gradual convergence to a near-optimal solution. SA performs a sequence of moves from a current solution to a better one according to specific transition rules while occasionally accepting some uphill solutions in order to guarantee diversity in the domain exploration and to avoid getting caught at local optima. The optimization process is managed by a cooling schedule that controls the number of iterations [1]. Thus, SA is useful to find the combination of the number of dendrites per class that results in the best classification rate in a finite number of iterations.

Algorithm 2 shows the pseudocode for DSN tuning with the SA algorithm. In line 3, the number of dendrites per class is randomly initialized in the range $[1, \sqrt{N_j}]$, where N_j is the number of patterns in the jth class. In line 10, the neighborhood structure generates a new solution by randomly moving (or not) backward or forward the number of dendrites per class. In lines 13–16, a DSN solution is accepted if its accuracy is higher than the previous solution; otherwise, a probability of acceptance criterion is applied, which depends on the current temperature. With this scheme, the current solution may be accepted even if it is worse than the previous solution, which is useful to avoid local optima.

Algorithm 2: DSN tuning based on simulated annealing.

Input: Training set $(\mathbf{X}, \mathbf{t})$; validation set $(\tilde{\mathbf{X}}, \tilde{\mathbf{t}})$; number of classes C
Output: Best solution $\mathbf{z}^*$

1 Set initial temperature, T_0
2 $t \leftarrow 0$
3 Create randomly an initial solution, $\mathbf{z}_0 = [l_1, \ldots, l_C]$
4 Train DSN with $\mathbf{z}_0$ and training set $(\mathbf{X}, \mathbf{t})$ `// Algorithm 1`
5 Evaluate the accuracy $f(\mathbf{z}_0)$ with validation set $(\tilde{\mathbf{X}}, \tilde{\mathbf{t}})$
6 Best solution, $\mathbf{z}^* \leftarrow \mathbf{z}_0$
7 **do**
8 $t \leftarrow t + 1$
9 $T_t \leftarrow 0.9 \cdot T_{t-1}$
10 Generate random solution $\mathbf{z}$ from the neighborhood $\mathcal{N}(\mathbf{z}_{t-1}) = \mathbf{z}_{t-1} + \mathbf{r}_t$, where $r_l \in \{-1, 0, 1\}$
11 Train DSN with $\mathbf{z}$ and training set $(\mathbf{X}, \mathbf{t})$ `// Algorithm 1`
12 Evaluate the accuracy $f(\mathbf{z})$ with validation set $(\tilde{\mathbf{X}}, \tilde{\mathbf{t}})$
13 **if** $f(\mathbf{z}) > f(\mathbf{z}_{t-1})$ **then**
14 $\mathbf{z}_t \leftarrow \mathbf{z}$
15 **else if** $\mathcal{U}(0, 1) \leq \exp\left(\frac{f(\mathbf{z}) - f(\mathbf{z}_{t-1})}{kT_t}\right)$ **then**
16 $\mathbf{z}_t \leftarrow \mathbf{z}$
17 **if** $(f(\mathbf{z}_t) > f(\mathbf{z}^*)) \vee ((f(\mathbf{z}_t) = f(\mathbf{z}^*)) \wedge (\sum_c \mathbf{z}_t < \sum_c \mathbf{z}^*))$ **then**
18 $\mathbf{z}^* \leftarrow \mathbf{z}_t$
19 **until** *cooling condition is reached*
20 **return** $\mathbf{z}^*$

Finally, in lines 16–17, the best solution is updated if its accuracy is lower than the current solution, or if both solutions have the same accuracy and the current solution has less number of dendrites than the current best solution.

4 Experiments

For evaluating the classification performance of the DSN approach, synthetic and real-world datasets are considered. The former comprises three didactic 2D datasets for illustrating the nonlinear boundaries generated by a DSN trained with Algorithm 2. On the other hand, ten real-world datasets were obtained from the UCI Machine Learning Repository [5], whose characteristics are summarized in Table 1. These datasets were also previously used to evaluate DMN, and DSN approaches [2,9].

For comparison purposes, the real-world datasets are also classified by MLP with one hidden layer, DMN initialized with the dHpC method and trained with SGD [9], and DEN trained with a hill-climbing algorithm for determining the number of dendrites per class [2]. In order to find statistical differences between

Table 1. Real-world datasets and their characteristics: identifier (ID), number of instances (N), number of classes (C), and dimensionality (D).

ID	Dataset	N	C	D
D_1	Breast cancer wisconsin	569	2	30
D_2	Glass identification	214	6	10
D_3	Heart dIsease cleveland	297	2	13
D_4	Hepatitis	112	2	18
D_5	Iris data	150	3	4
D_6	Page blocks	5409	5	10
D_7	Pima Indians diabetes	768	2	8
D_8	Seeds	199	3	7
D_9	Thyroid gland data	215	3	5
D_{10}	Wine recognition data	178	3	13

methods, the Kruskal–Wallis test ($\alpha = 0.05$) is used to evaluate whether the medians of the approaches compared differ under the assumption that the shapes of the underlying distributions are the same. Also, the correction for multiple testing on the basis of the same data is made by the Bonferroni method.

The k-fold cross-validation method (with $k = 10$) is used to built training and test sets to measure the classification accuracy (i.e., the hit rate) of neural models. Moreover, in Algorithm 2, the training set is partitioned again into two parts to create the training (80%) and validation (20%) sets.

It is worth mentioning that a procedure of grid search and k-fold cross-validation (with $k = 5$) determines the number of hidden neurons that maximizes the accuracy of the MLP network, where the number of hidden neurons is increased from 5 to 100 neurons, in steps of 5 [4].

5 Results

Figure 3 shows the distribution of hyperspheres per class (i.e., dendrites) for each synthetic dataset. The SA algorithm determined the number of dendrites that maximized accuracy. For instance, it is notable that only three dendrites (one per class) are required for correctly classifying all the patterns of the Ring dataset (Fig. 3(b)). The corresponding decision regions obtained by the DSN approach are also illustrated in Fig. 3. Notice that nonlinear decision boundaries are built, which are capable of modeling complex class distributions.

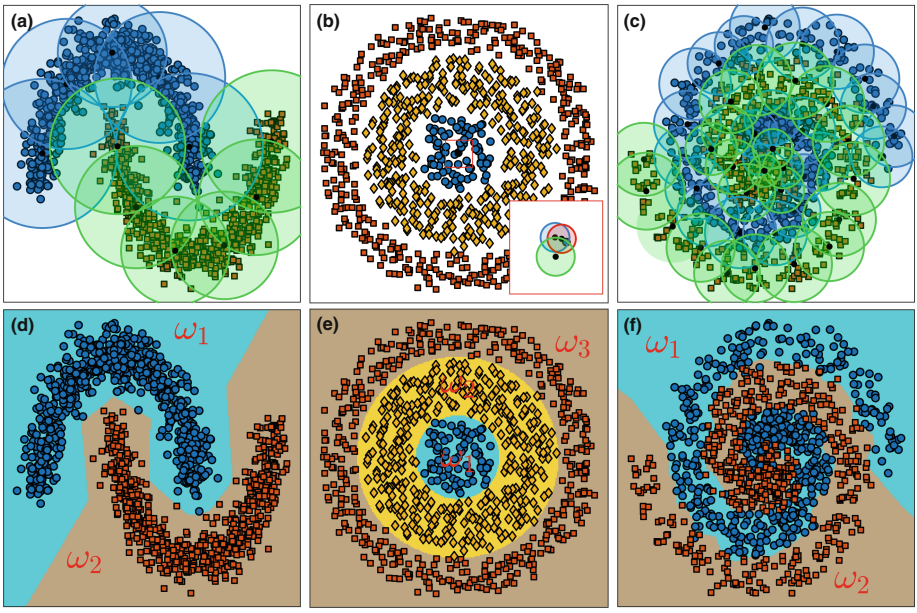

Fig. 3. Top row, the distribution of hyperspheres for the 2D synthetic datasets. The number of dendrites is: (a) Horseshoes: 11, (b) Rings: 3 (with zoom to view hyperspheres), and (c) Two-spirals: 46. Bottom row, the decision regions generated by the DSN approach. The classes are represented by ω_1 (class 1), ω_2 (class 2), and ω_3 (class 3). The accuracy measured on the validation set is: (d) Horseshoes: 100%, (e) Rings: 100%, and (f) Two-spirals: 93%.

In the case of real-world datasets, Fig. 4 shows the accuracy results obtained by MLP, DMN, DEN, and DSN neural models. It is remarkable that DSN outperformed the DEN approach in eight of ten datasets, and obtained competitive results in relation to DMN and MLP methods. Moreover, the multiple comparisons with the Kruskal–Wallis test and Bonferroni correction determined that DSN did not present statistically significant differences with MLP ($p = 0.7035$) and DMN ($p = 0.3037$), whereas DSN and DEN were statistically significantly different ($p < 0.0001$).

In addition, for all the datasets, the DSN presented a simpler structure than MLP and DEN. For instance, for the Thyroid Gland dataset (D_9), MLP obtained an accuracy of 97.2% with 41 hidden neurons, whereas DSN reached an accuracy of 96.8% with four dendrites. Also, for the Page Blocks dataset (D_6), the accuracy of DEN is 91.1% with 71 dendrites, whereas DSN used 59 dendrites to attain an accuracy of 93.8%.

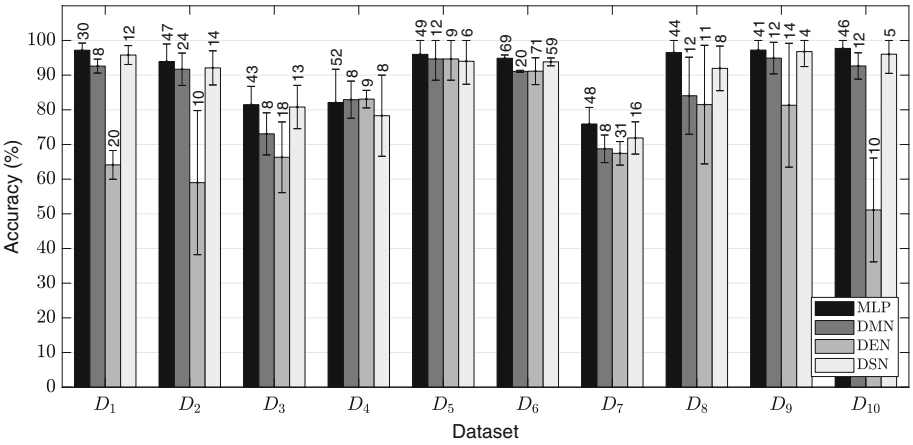

Fig. 4. Accuracy results of neural models MLP, DMN, DEN, and DSN for real-world datasets. The height of the bars represents the mean of 10-folds of cross-validation. The error bars are the standard deviations. The numerical values in the top of bars denote the average number of hidden neurons or dendrites for each neural model.

6 Discussion and Conclusions

In this paper, it was presented the theoretical basis of the Dendrite Spherical Neurons (DSN) for pattern classification. The DSN can be categorized in the family of neuronal models with dendritic processing, like Dendrite Morphological Neurons (DMN) and Dendrite Ellipsoidal Neurons (DEN). These neural models are an alternative to the Multilayer Perceptron (MLP) to solve classification problems with a simple architecture. Moreover, the DSN model can be viewed as a simplification of the DEN model, whose covariance matrix is diagonal with all its elements equal. DSN can overcome potential issues found in DEN, such as singular covariance matrices when a dendrite clusters a small number of patterns, while maintaining the smoothness of decision boundaries.

The number of dendrites in DSN is a free parameter that should be tuned adequately. Thus, we proposed an optimization procedure based on the Simulated Annealing (SA) algorithm, in which the classification accuracy is maximized, while the number of dendrites is used as a constraint. Notice that this scheme does not guarantee the minimum number of dendrites, which represents a limitation of the proposed method. Hence, the problem of DSN tuning can be further extended to multiobjective optimization, in which the accuracy is maximized, and the number of dendrites is minimized.

The experiments with real-world datasets revealed that DSN tended to reach better accuracy results than DMN. This behavior is because the DMN strongly depends on its initial solution (here was used the dHpC method) that is refined by Stochastic Gradient Descent (SGD); therefore, the error obtained in the initial solution will be carried to the final solution. Unlike to DMN, DSN does not refine an initial solution but uses the k-means algorithm to distribute the dendrites

over the input space, while the SGD is used to train the weights of the softmax function at the output of the neuron.

On the other hand, DEN obtained the lowest classification performance. This behavior is because dendrites are created independently for each class without considering the interaction between dendrites of different classes, causing overlaps in transition regions between classes. This drawback is addressed in the DSN model by considering the closeness between dendrites for calculating the radius of the hyperspheres.

DSN obtained competitive results concerning MLP for classifying real-world datasets. However, MLP obtained more complex structures than DSN; that is, MLP usually requires more hidden neurons than dendrites in DSN to model the same classification problem. Therefore, DSN can be potentially used for classification problems where computational resources are limited.

Future work involves a study of the effect of the number of dendrites on the DSN classification performance. Also, an extensive study with larger datasets and other kinds of classifiers is pending. Besides, the accuracy of DNS can be improved by using other distance metrics to measure the closeness between patterns as well as applying to SDG mechanisms of momentum and adaptive learning rate.

Acknowledgements. We want to express our sincere appreciation to CINVESTAV-IPN and the Instituto Politécnico Nacional for the support provided to carry out this research. This research was supported by a Fondo SEP-Cinvestav 2018 grant (No. 145), and SI-IPN 20190007 and SIP-IPN 20200630.

References

1. Amine, K.: Multiobjective simulated annealing: principles and algorithm variants. J. Multivar. Anal. **143**, 1–13 (2019)
2. Arce, F., Zamora, E., Fócil-Arias, C., Sossa, H.: Dendrite ellipsoidal neurons based on k-means optimization. Evol. Syst. **10**(3), 381–396 (2018). https://doi.org/10.1007/s12530-018-9248-6
3. Arce, F., Zamora, E., Sossa, H., Barrón, R.: Differential evolution training algorithm for dendrite morphological neural networks. Appl. Soft Comput. **68**, 303–313 (2018)
4. Barbero-Jimenez, A., Lopez-Lazaro, J., Dorronsoro, J.R.: Finding optimal model parameters by deterministic and annealed focused grid search. Neurocomputing **72**(13), 2824–2832 (2009)
5. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
6. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley, New York (2012)
7. Priddy, K.L., Keller, P.E.: Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering), vol. TT68. SPIE- International Society for Optical Engineering, Bellingham (2005)

8. Sossa, H., Arce, F., Zamora, E., Guevara, E.: Morphological neural networks with dendritic processing for pattern classification. In: Vergara Villegas, O.O., Nandayapa, M., Soto, I. (eds.) Advanced Topics on Computer Vision, Control and Robotics in Mechatronics, pp. 27–47. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77770-2_2
9. Zamora, E., Sossa, H.: Dendrite morphological neurons trained by stochastic gradient descent. Neurocomputing **260**, 420–431 (2017)