Lattice Algebra Approach to Single-Neuron Computation

Gerhard X. Ritter, *Senior Member, IEEE*, and Gonzalo Urcid

***Abstract*—Recent advances in the biophysics of computation and neurocomputing models have brought to the foreground the importance of dendritic structures in a single neuron cell. Dendritic structures are now viewed as the primary autonomous computational units capable of realizing logical operations. By changing the classic simplified model of a single neuron with a more realistic one that incorporates the dendritic processes, a novel paradigm in artificial neural networks is being established. In this work, we introduce and develop a mathematical model of dendrite computation in a morphological neuron based on lattice algebra. The computational capabilities of this enriched neuron model are demonstrated by means of several illustrative examples and by proving that any single layer morphological perceptron endowed with dendrites and their corresponding input and output synaptic processes is able to approximate any compact region in higher dimensional Euclidean space to within any desired degree of accuracy. Based on this result, we describe a training algorithm for single layer morphological perceptrons and apply it to some well-known nonlinear problems in order to exhibit its performance.**

***Index Terms*—Artificial neural networks, dendrite computation, lattice algebra, morphological neural networks, perceptron training, single-neuron computation.**

I. Introduction

RECENT advances in neurobiology and the biophysics of neural computation have brought to the foreground the importance of dendritic structures of neurons. These structures are now viewed as the primary basic computational units of the neuron, capable of realizing logical operations. Based on these new biophysical neural models we develop a new paradigm for computing with morphological neurons that incorporates the dendritic processes, thus yielding a more realistic model of single neuron computation in artificial neural networks.

Since the computational units and the topology of the proposed model mimic those biological neurons of the cerebral cortex, we begin by providing a brief review of biological neurons and their processes in Section II. In Section III, we provide a short introduction to lattice algebra and morphological neural computation. In Section IV, the model of morphological neural computation is generalized to include the dendritic processes described in Section II. Several examples are also given in order to illustrate the computational processes and capabilities of a morphological neuron with dendritic structure. Section V establishes the core mathematical result concerning the computational capability of a morphological neuron with dendrites, namely that any compact region of n-dimensional Euclidean space, $\mathbb{R}^n$, can be approximated to any given degree of accuracy with a single neuron. Based on the proof of this result (which is presented in the Appendix) we developed a training algorithm that is described in Section VI. To exhibit the performance of the training algorithm, we include some applications to well-known nonlinear problems. We conclude this paper with several pertinent observations.

Manuscript received May 30, 2002. This work was supported in part by Frontier Technology Inc. under U.S. Army Project DACA41-02-C-007. The work of G. Urcid was supported in part by a postdoctoral scholarship (110372) from Consejo Nacional de Ciencia y Tecnología (CONACyT) in Mexico.

G. X. Ritter is with the CISE Department, University of Florida, Gainesville, FL 32611 USA (e-mail: ritter@cise.ufl.edu).

G. Urcid is with the Optics Department, INAOE, Tonantzintla, Pue. 72000, Mexico (e-mail: gurcid@inaoep.mx).

Digital Object Identifier 10.1109/TNN.2003.809427

II. Neurons and Their Processes

To a certain extent, this paper is concerned with the confluence of two streams of research: the study of the computational organization of biological neurons and the study of the computational capabilities of artificial neurons based on lattice algebra. Single morphological neuron computation takes into account input and output synaptic responses as well as computation in dendrites. In this sense, morphological neural computation mimics the biological model. Owing to the close coupling of these two models, it will be useful to provide a brief review of some essential features of biological neurons. These features are used to accommodate a more realistic model of morphological neurons.

The *nerve cell* or *neuron* is the structural unit of the nervous system. It consists of a cell body, called *soma*, and several processes. These processes are of two kinds and are called, respectively, *dendrites* and *axons*. The dendrites, which are usually multiple, conduct impulses toward the body of the cell; the axon conducts from the cell body. Dendrites typically have many branches that create large and complicated trees. Many (but not all) types of dendrites are studded with large numbers of tiny branches called *spines*. Dendritic spines, when present, are the major *postsynaptic* target for *excitatory synaptic* input. The soma and the dendrites constitute the input surface of the neuron.

The axon which usually arises from the opposite pole of the cell at a point called the *axon hillock*, consists of a long fiber whose branches form the *axonal arborization* or *axonal tree*. For some neurons, the axon may have branches at intervals along its length in addition to its terminal arborization. The tips of the branches of the axon are called *nerve terminals* or *boutons* or *synaptic knobs*. The axon is the principal fiber branch of the neuron for the transmission of signals to other neurons. Fig. 1 provides an illustration of a typical neuron with its branching processes. An impulse traveling along an axon from the axon

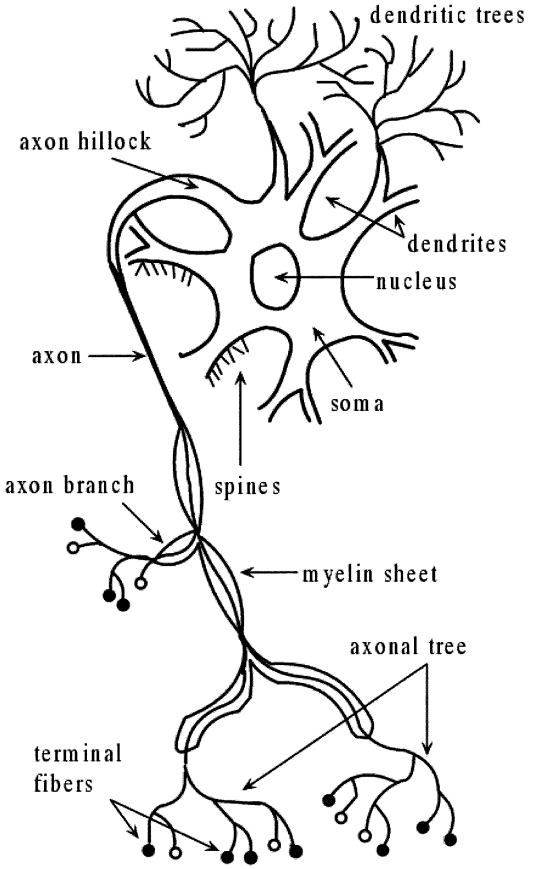

Fig. 1. Diagram of a neuron cell showing dendrites, dendritic trees, axon branches, and terminal branches. Excitatory and inhibitory inputs are indicated, respectively, by black small disks (•) and small circles (∘).

hillock propagates through the axonal tree all the way to the nerve terminals. The terminals of the branches make contact with the soma and the many dendrites of other neurons. The sites of contact are the *synaptic sites* where the *synapses* take place. The *synapse* is a specialized structure whereby neurons communicate but there is no actual structural union of the two neurons at the synaptic site. The synaptic knob is separated from the surface of the dendrite or soma by an extremely narrow space called the *synaptic cleft*. The exact mechanism of synaptic structures is fairly well understood and it is well-known that there exist two kinds of synapses; *excitatory synapses* which tend to depolarize the postsynaptic membrane, and, consequently, play a role in exciting the postsynaptic cell to fire impulses, and *inhibitory synapses* that try to prevent the neuron from firing impulses in response to excitatory synapses. Inhibitory action affects the postsynaptic membrane and lowers its potential [15], [45].

The number of synapses on a *single* neuron in the cerebral cortex ranges between 500 to 200 000. Most of the synapses occur on the dendritic tree of the neuron, and it is here where information is processed [22], [32], [44], [45]. Dendrites make up the largest component in both surface area and volume of the brain. Part of this is due to the fact that pyramidal cell dendrites span all cortical layers in all regions of the cerebral cortex [15], [22], [44]. Thus, when attempting to model artificial brain networks that bear more than just a passing resemblance to biological brain networks, one cannot ignore dendrites (and their associated spines) which can make up more than 50% of the neuron's membrane. This is especially true in light of the fact that some brain researchers have proposed that dendrites and not the neuron are the elementary computing devices of the brain. Neurons with dendrites can function as many, almost independent, functional subunits with each unit being able to implement a rich repertoire of logical operations, [22], [27], [32], [44], [54]. Possible mechanisms for dendritic computation of such logical functions as XOR, AND, and NOT have been proposed by several researchers [22], [32], [28], [21], [44], [49]. For a more thorough background in dendritic computing, we refer the reader to [1], [15], [22], and [26].

III. Computational Basis for Morphological Neurons

In recent years, operations based on lattice algebra have found widespread applications in the engineering sciences. In these applications, the usual arithmetic operations of addition and multiplication are replaced by corresponding lattice operations. Lattice induced operations lead to an entirely different perspective of a class of nonlinear transformations. These ideas were applied by Shimbel [50] to communications networks, and to machine scheduling by Cuninghame-Green [5], [6] and Giffler [17]. Others have discussed their usefulness in applications to shortest path problems in graphs [29]. Additional examples are given in [7], primarily in the field of operations research. Application to image processing were first developed by Ritter and Davidson [8], [33], [34]. These applications were successfully extended to artificial neural networks [9], [10], [12]–[14], [51], [55], [56]. Artificial neural networks whose computation are based on lattice algebra have become known as *morphological neural networks* (MNNs). The primary distinction between traditional neural networks and MNNs is the computation performed by the individual neuron. Traditional neural networks use a multiply accumulate neuron with thresholding over the ring $(\mathbb{R}, +, \times)$ given by

$$\tau_j(\mathbf{x}) = \sum_{i=1}^{n} x_i w_{ij} - \theta_j \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$, x_i denotes the value of the ith neuron, w_{ij} denotes the synaptic strength between the ith neuron and the jth neuron, θ_j the threshold of the jth neuron, and τ_j the total input to the jth neuron. Morphological neural networks use lattice operations $\vee$ (maximum), or $\wedge$ (minimum), and $+$ from the semirings $(\mathbb{R}_{-\infty}, \vee, +)$ or $(\mathbb{R}_{\infty}, \wedge, +)$ where $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$ and $\mathbb{R}_{\infty} = \mathbb{R} \cup \{\infty\}$. The computation at a neuron in a MNN for input $\mathbf{x} = (x_1, \ldots, x_n)$ is given by

$$\tau_j(\mathbf{x}) = p_j \bigvee_{i=1}^{n} r_{ij}(x_i + w_{ij}) \tag{2}$$

or

$$\tau_j(\mathbf{x}) = p_j \bigwedge_{i=1}^{n} r_{ij}(x_i + w_{ij}) \tag{3}$$

where $r_{ij} = \pm 1$ denotes whether the ith neuron causes excitation or inhibition on the jth neuron, and $p_j = \pm 1$ denotes the output response (excitation or inhibition) of the jth neuron to

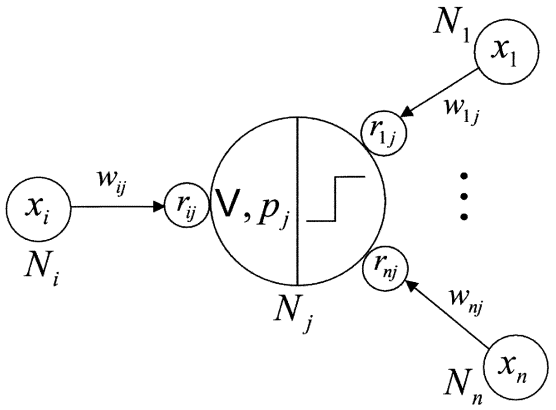

Fig. 2. Computational model of a morphological neuron.

the neurons whose axons contact the jth neuron. For excitatory input and output action, $r_{ij} = 1$ and $p_j = 1$, respectively. Inhibitory input and output responses have the values $r_{ij} = -1$ and $p_j = -1$, respectively. The computational model for a neuron in a MNN that uses the maximum operator is illustrated in Fig. 2. The activation function $f(\tau_j)$ in a MNN is generally a hard limiter.

There are several advantages for replacing the arithmetic operations used in traditional neural computing with lattice based operations. It is apparent from (2) and (3) that morphological neural computation does not involve multiplications but only the operations of OR or AND, addition, and subtraction. This provides for extremely fast neural computation and easy hardware implementation. Convergence problems and lengthy training algorithms are often nonexistent [37], [39]. It has been shown that morphological associative memories are extremely robust in the presence of noise and have unlimited storage capacity [36], [38], [40], [41][53]. There is a close, natural connection between lattice-based computing and fuzzy set theory making lattice based networks amenable to handling more general data types and particular types of learning models [30]. Finally, morphological neural networks are capable of solving any conventional computational problem and, thus, can be inherently useful in a wide variety of application domains [35].

In this paper, we shall use the lattice algebra $(\mathbb{R}_\infty, \wedge, +)$ for our underlying neural computation. This algebra obeys the following laws: if $a, b, c \in \mathbb{R}_\infty$, then

$$\begin{aligned} a + (b \wedge c) &= (a + b) \wedge (a + c) \\ a \wedge \infty &= \infty \wedge a = a \\ a + \infty &= \infty + a = \infty \\ a + 0 &= 0 + a = a. \end{aligned} \tag{4}$$

To compare this system with the system $(\mathbb{R}, +, \cdot)$ used in traditional neural computing, we have the analogous laws, if $a, b, c \in \mathbb{R}$, then

$$\begin{aligned} a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\ a + 0 &= 0 + a = a \\ a \cdot 0 &= 0 \cdot a = 0 \\ a \cdot 1 &= 1 \cdot a = a. \end{aligned} \tag{5}$$

Thus, in the system $(\mathbb{R}_\infty, \wedge, +)$, the symbol ∞ acts as the *null element* and 0 as the *unit element* if we view $\wedge$ as addition and $+$ as multiplication.

In addition to the laws given in (4) and (5), the two systems share many other properties such as commutativity and associativity. In this sense, the two systems are mathematically very similar and, therefore, one could be easily substituted for the other in many application areas. However, there are several differences, a major one is the lack of inverses for the lattice operation of minimum, namely $\wedge$. It is this lattice operation that provides for the unique and distinct properties of morphological neural networks.

It must also be mentioned that Koch and Poggio make a strong case for multiplying with synapses, i.e., for $x_i \cdot w_{ij}$ [23]. In the model presented here, multiplication could have been used just as well. A mathematical equivalent theory can be obtained by replacing the lattice $(\mathbb{R}_\infty, \wedge, +)$ by the lattice $(\mathbb{R}^{\geqslant 0}, \vee, \cdot)$ where $\mathbb{R}^{\geqslant 0} = \{x \in \mathbb{R}: x \geqslant 0\}$. The mathematical equivalence is given by the isomorphism $\varphi: \mathbb{R}_\infty \to \mathbb{R}^{\geqslant 0}$ defined by $\varphi(x) = e^{-x}, \forall x \in \mathbb{R}$, and $\varphi(\infty) = 0$. It is easy to see that φ is a bijection. Furthermore, $\varphi(x + y) = \varphi(x) \cdot \varphi(y)$ and $\varphi(x \wedge y) = \varphi(x) \vee \varphi(y)$. Therefore, φ preserves the algebraic structure and is an isomorphism; note also that $\varphi(0) = 1$. Hence, identities are mapped to identities. The inverse of φ, $\varphi^{-1}: \mathbb{R}^{\geqslant 0} \to \mathbb{R}_\infty$, is given by $\varphi^{-1}(x) = -\ln x$ for $x > 0$ and $\varphi^{-1}(0) = \infty$. This proves that, mathematically, nothing is gained by replacing the additive lattice with the multiplicative lattice. Although a multiplicative lattice is more intuitive from a biological standpoint, a reason for using additive synapses is the reduction of computational overhead and ease of describing the proposed (morphological) dendritic model of computation. Additionally, the additive lattice is directly related to mathematical morphology, a topic well known to the image processing and computer vision community [11], [18]. We realize that some researchers are not too familiar with the notion and uses of algebraically equivalent structures. In particular, questions may be raised pertaining to the meaning and use of negative weights. Negative weights arise quite naturally when using the additive lattice $(\mathbb{R}_\infty, \wedge, +)$. Traditionally, *weight* is a positive quantity with the sign determining whether the input is excitatory or inhibitory. Negative weights have nothing to do with inhibition in the additive lattice. In the additive lattice it is the sign of r_{ij} that determines excitatory or inhibitory input. For a biological (or traditional) interpretation of negative weights, simply observe that these correspond to large positive weights in the equivalent multiplicative lattice $(\mathbb{R}^{\geqslant 0}, \vee, \cdot)$ as $w = e^{-w_{ij}} > 0 \, \forall \, w_{ij} \in \mathbb{R}$ and $e^{-x} \to \infty$ as $x \to -\infty$.

The lattice $(\mathbb{R}_\infty, \wedge, +)$ has a natural dual (isomorphic) structure given by $(\mathbb{R}_{-\infty}, \vee, +)$, where, $\forall x \in \mathbb{R}_{-\infty}$, $x + (-\infty) = (-\infty) + x = -\infty$. For each $x \in \mathbb{R}_{\pm\infty} = \mathbb{R} \cup \{-\infty, \infty\}$ we define its *dual* or *conjugate* by $x^* = -x$ where $-(-\infty) = \infty$. The following duality laws are a direct consequence of this definition:

$$\begin{aligned} (x^*)^* &= x \\ (x \wedge y)^* &= x^* \vee y^* \\ (x \vee y)^* &= x^* \wedge y^*. \end{aligned} \tag{6}$$

The function ψ: $\mathbb{R}_{-\infty} \rightarrow \mathbb{R}_{\infty}$ defined by $\psi(x) = x^*$ is an isomorphism. Therefore, a morphological neural net using the operation $\vee$ can always be reformulated in terms of using the operation $\wedge$, and vice versa. Due to the relations $-(x \wedge y) = -x \vee -y$ and $-x \wedge -y = -(x \vee y)$ of (6), and the use of inhibitory input and output, the algebra $(\mathbb{R}_{-\infty}, \vee, +)$ is implicitly incorporated in our computational model which is defined in the next section [see (7) and (8)].

IV. Model of Dendritic Computation in a Morphological Neuron

A morphological neuron N has distinct dendritic postsynaptic regions that receive input from terminal axonal branches of other neurons. The knobs of the terminal branches provide either excitatory or inhibitory input, while the postsynaptic membrane of the dendrites or soma determines excitatory or inhibitory response to the received input. The neuron responds to the *total* dendritic input and is enabled by an activation function f.

If N receives input from the n neurons $N_1, N_2, \ldots, N_n$, then each N_i has axonal branches that terminate at various synaptic regions of N. In the proposed model these regions are distributed along a finite number of dendrites $D_1, \ldots, D_K$. Each neuron N_i will have *at most* two axonal branches terminating on a given dendrite D_k of N. If two axonal branches of N_i terminate at the same dendrite D_k, then one branch is assumed to provide an inhibitory input and the other an excitatory input to the dendrite. The reasons for these assumptions are based on the properties of the lattice algebra $(\mathbb{R}_{\infty}, \wedge, +)$ since inputs to the dendrites will be minimized or maximized. Therefore, having more than one excitatory or inhibitory axonal branch from one neuron terminating on the same dendrite of N will not provide for additional computational advantages. This fact can also be ascertained from the subsequent discussion and examples. It is important to emphasize that the assumption that one branch of the neuron N_i can be excitatory and another inhibitory to dendrites of the same neuron diverges from biological fact. Real neurons do not provide for this capability.

The synaptic strength of the axonal branch of neuron N_i terminating on the kth dendrite of N will be denoted by w_{ik}^l, where $l \in \{0, 1\}$. The superscript l distinguishes between an excitatory and inhibitory branch. In particular w_{ik}^1 denotes the fact that the terminal of the fiber will induce excitation, while w_{ik}^0 will induce inhibition. The input response of the branch fiber w_{ik}^l will be denoted by $r_{ik}^l = \pm 1$, where $r_{ik}^1 = +1$ denotes excitation and $r_{ik}^0 = -1$ denotes inhibition. Since $r_{ik}^l = (-1)^{(1-l)}$, where $(-1)^0 \equiv 1$, it will simplify notation to replace r_{ik}^l with $(-1)^{(1-l)}$ whenever it is obvious as to which branch of a neuron N_i terminates at $(-1)^{(1-l)}$. The postsynaptic membrane of the kth dendrite of N responds to the minimum of the total input received and tries to either inhibit or accept the received input. We also assume that all inputs have a *finite* value, i.e., $\forall i \in \{1, \ldots, n\},\ x_i \neq \infty$. Fig. 3 provides a graphical representation of this model.

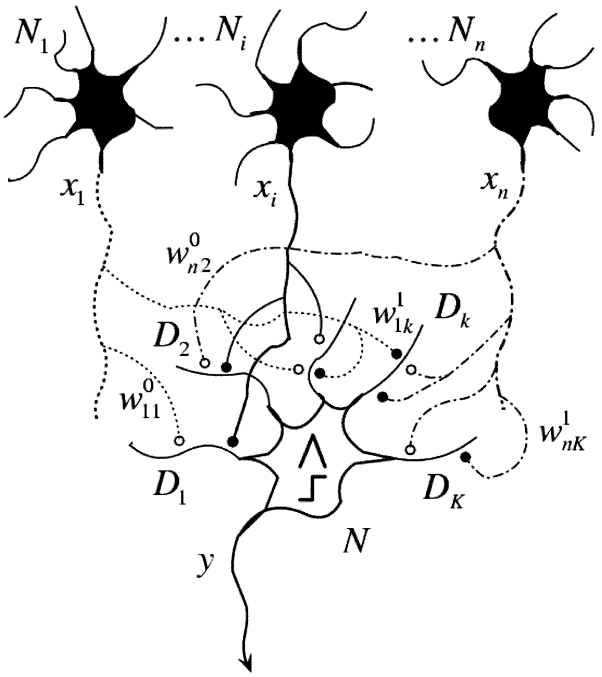

Fig. 3. Morphological neuron with dendrites. Input neuron N_i can synapse dendrite k of N with excitatory or inhibitory fibers, e.g., w_{1k}^1 is the weight of an excitatory fiber coming from input neuron N_1 into dendrite D_k, while w_{n2}^0 is the weight of an inhibitory fiber from neuron N_n into dendrite D_2.

The computation performed by the kth dendrite for input $\mathbf{x} \in \mathbb{R}^n$ is given by

$$\begin{aligned} \tau_k(\mathbf{x}) &= p_k \bigwedge_{i \in I} \bigwedge_{l \in L} r_{ik}^l (x_i + w_{ik}^l), \\ &= p_k \bigwedge_{i \in I} \bigwedge_{l \in L} (-1)^{(1-l)} (x_i + w_{ik}^l). \end{aligned} \tag{7}$$

Here I corresponds to the set of all input neurons N_i with terminal fibers that synapse on the kth dendrite of N, while L corresponds to the set of terminal fibers of the ith neuron that synapse on the kth dendrite of N, and $p_k \in \{-1, 1\}$ denotes the excitatory or inhibitory response of the kth dendrite. Note that the set $I \subseteq \{1, \ldots, n\}$ can be different for each k and the set $L \subseteq \{0, 1\}$ also may be distinct for each $i \in I$ and k. Clearly, $I \neq \emptyset$ and $L \neq \emptyset$ since there is at least one axonal fiber coming from at least one of the input neurons with synapse on dendrite k.

The value $\tau_k(\mathbf{x})$ is passed to the cell body and the state of the neuron N is a function of the inputs received from all its dendrites. That is

$$\tau(\mathbf{x}) = \bigwedge_{k=1}^{K} \tau_k(\mathbf{x}) \tag{8}$$

where K denotes the total number of dendrites (synaptic input regions) of N. The *next* state of N is determined by an activation function f. In this exposition we will restrict our discussion to the hard limiter

$$f(\tau) = \begin{cases} 1, & \text{if } \tau \geq 0 \\ 0, & \text{if } \tau < 0. \end{cases} \tag{9}$$

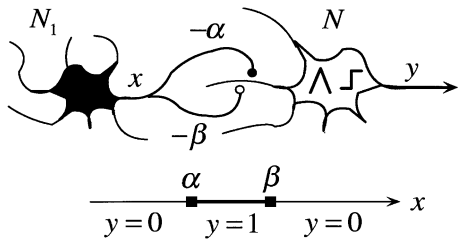

Fig. 4. Input neuron N_1 provides input $x \in \mathbb{R}$ to the neuron N and only values from the closed interval $[\alpha, \beta]$ will fire N.

The total state computation of neuron N is, therefore, given by

$$y(\mathbf{x}) = f\left(\bigwedge_{k=1}^{K}\left[p_k \bigwedge_{i\in I}\bigwedge_{l\in L}(-1)^{(1-l)}(x_i + w_{ik}^l)\right]\right). \quad (10)$$

Having defined the computational model of a morphological neuron with dendritic structure, it will be instructive to provide a few simple examples in order to illustrate some basic properties of this model.

Example 1: The simplest case occurs when N receives input from a single *input* neuron N_1 having only one terminal axonal branch with a synapse on N and only excitatory responses. As an additional assumption we also suppose that the input neuron is allowed to send values from the domain $\mathbb{R}$ down its axon. If $w = -\alpha$ denotes the strength of the synaptic connection, then any value $x \in \mathbb{R}$ will result in an *on* state of N if and only if $x \geq \alpha$ since

$$\tau(x) = x + w \geq 0 \Leftrightarrow x \geq -w = \alpha. \quad (11)$$

In geometric terms, the neuron N responds to all points in the positive direction of the x-axis that are not less than $-w$. The neuron N can be *inhibited* from *seeing* all points on the interval $[\alpha, +\infty)$ by imposing input inhibition. For instance, if $\alpha < \beta$, defining two terminal axonal branches for N_1 with one having weight $w_1 = -\alpha$ and response $r_1 = +1$, and the other having weight $w_2 = -\beta$ and response $r_2 = -1$, then

$$\tau(x) = r_1(x + w_1) \wedge r_2(x + w_2) \geq 0 \Leftrightarrow x \in [\alpha, \beta]. \quad (12)$$

Hence, only points in the interval $[\alpha, \beta]$ will fire N. Fig. 4 provides a graphical representation of this particular example. It is interesting to note that if input x is interpreted as the "firing frequency" of neuron N_1 and the weights w, w_1, w_2 are interpreted as firing frequency "thresholds," then the output neuron N acts as a filtering device. Consequently, the two cases discussed illustrate, respectively, high-pass and band-pass filtering.

In the preceding example, we have $\tau(x) = p\bigwedge_{i=1}^{2} r_i(x+w_i)$, where $p = 1$. Reversing the output response is equivalent to interchanging the *on* and *off* regions of N since

$$-\bigwedge_{i=1}^{2} r_i(x + w_i) = -[(x-\alpha) \wedge -(x-\beta)] \geq 0 \quad (13)$$

if and only if $x \leq \alpha$ or $x \geq \beta$. It is noteworthy to observe that the *boundary points* $x = \alpha$ and $x = \beta$ remain *on* points for N. Expanding on this simple example, it is easy to ascertain that for a single input neuron N_1, the neuron N is capable of identifying any finite number of points and intervals on the line $\mathbb{R}$.

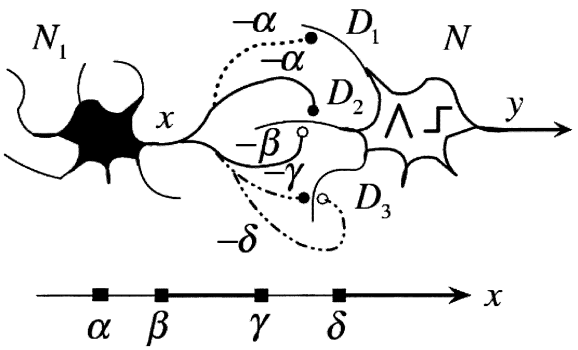

Fig. 5. Neuron N will fire $(y = 1)$ for input values from the set $X = \{\alpha\} \cup [\beta, \gamma] \cup [\delta, +\infty)$. If $x \in \mathbb{R} - X$ then $y = 0$.

TABLE I
WEIGHTS AND SYNAPTIC RESPONSES, EX.2

D_k	w_{1k}^1	r_{1k}^1	w_{1k}^0	r_{1k}^0	p_k
D_1	$-\alpha$	$+1$	$-\infty$	-1	$+1$
D_2	$-\alpha$	$+1$	$-\beta$	-1	-1
D_3	$-\gamma$	$+1$	$-\delta$	-1	-1

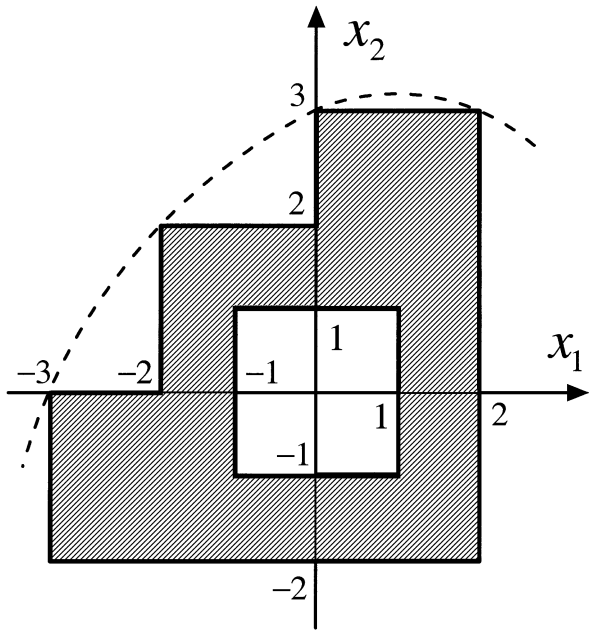

Fig. 6. Two-dimensional connected nonconvex region X recognized by the single neuron N with four dendrites.

Example 2: If $\alpha < \beta < \gamma < \delta$, the neuron shown in Fig. 5 will fire (state $y = f(\tau(x)) = 1$) when receiving the inputs $x = \alpha$ or any value from the intervals $[\beta, \gamma]$ and $[\delta, +\infty)$ and will not fire (state $y = f(\tau(x))=0$) for all other values from $\mathbb{R}$. In this example, N has three dendrites D_1, D_2, and D_3, with corresponding terminal axonal branching weights and input, and output responses given in Table I. For algebraic consistency as well as numerical computation when using (10), *unused* terminal fibers with a hypothetical excitatory or inhibitory input will be assigned, respectively, a weight value of $+\infty$ or $-\infty$. Also, as explained earlier, $r_{1k}^1 = +1$ and $r_{1k}^0 = -1$ for

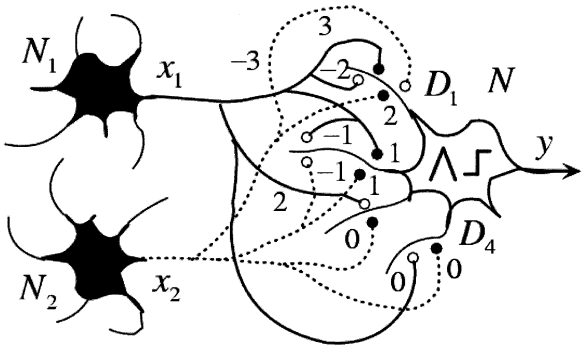

Fig. 7. Neuron N will fire $(y = 1)$ for input values from the region $X \subset \mathbb{R}^2$ depicted in Fig. 6; if $\mathbf{x} \in \mathbb{R}^2 - X$ then $y = 0$.

TABLE II
WEIGHTS AND POSTSYNAPTIC RESPONSES, EX.3

D_k	w^1_{1k}	w^1_{2k}	w^0_{1k}	w^0_{2k}	p_k
D_1	3	2	-2	-3	$+1$
D_2	1	1	-1	-1	-1
D_3	$+\infty$	0	2	$-\infty$	-1
D_4	$+\infty$	0	0	$-\infty$	-1

all $k \in \{1, \ldots, K\}$. Hence, in the following examples, we will omit these columns.

The preceding one-dimensional examples easily generalize to higher dimensions. We conclude this section with a few two-dimensional examples that will further illuminate properties of single morphological neuron computation.

Example 3: The *on* region of the neuron N in this example consists of all points $\mathbf{x} = (x_1, x_2)$ within the shaded area shown in Fig. 6. As illustrated in Fig. 7, the neuron N receives input from two input neurons N_1 and N_2, and its morphology is determined by four dendrites $D_1, \ldots, D_4$. The axonal terminal fibers of N_1 and N_2 have weights and output responses given in Table II.

Example 3 also illustrates how a region bounded by any type of surface (e.g., the dotted curve shown in Fig. 6) can be approximated via step functions as an *on* region for N to within any given degree $\varepsilon > 0$ of accuracy. As a final example, we consider three different solutions to the exclusive or problem XOR.

Example 4: For the classical XOR problem the set of possible input values is $X = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. Given an input vector $\mathbf{x} = (x_1, x_2) \in X$, the desired neural output is

$$f(\tau(\mathbf{x})) = \begin{cases} 1 \Leftrightarrow \mathbf{x} \in \{(0, 1), (1, 0)\} \\ 0 \Leftrightarrow \mathbf{x} \in \{(0, 0), (1, 1)\} \end{cases}. \tag{14}$$

The simple neural network shown in Fig. 8 solves this problem. Here N has two dendrites with excitatory output responses that receive input from the input neurons N_1 and N_2. The particular weights and output responses are summarized in Table III.

Example 5: The on region for the neuron N of Example 4 is the entire shaded region shown in Fig. 9 if we allow vectors from $\mathbb{R}^2 = \{(x_1, x_2)\colon x_i \in \mathbb{R}, i = 1, 2\}$ as input. Thus, N would classify all points in the shaded region to belong to the same class. Suppose, however, that we want only the points of $S = \{(0, 1), (1, 0)\}$ to be classified as belonging to class C_1 (i.e., $f(\tau(\mathbf{x})) = 1 \Leftrightarrow \mathbf{x} \in S$) and all other points as not belonging to class C_1; i.e., $f(\tau(\mathbf{x})) = 0 \Leftrightarrow \mathbf{x} \notin S$. To solve this problem, another dendrite needs to be added and the weight values need to be changed. The weights and output responses listed in Table IV define the network shown in Fig. 10 and provide a solution to the posed problem. The problem posed here is a classical *one class*

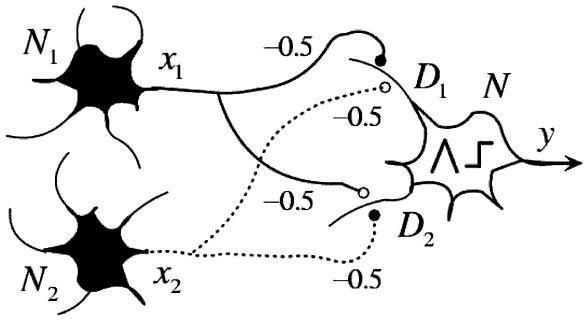

Fig. 8. Single morphological neuron solution to the XOR problem using two dendrites. Inputs are not restricted to set X.

TABLE III
XOR NET PARAMETERS (UNRESTRICTED INPUT), EX.4

D_k	w^1_{1k}	w^1_{2k}	w^0_{1k}	w^0_{2k}	p_k
D_1	-0.5	$+\infty$	$-\infty$	-0.5	$+1$
D_2	$+\infty$	-0.5	-0.5	$-\infty$	$+1$

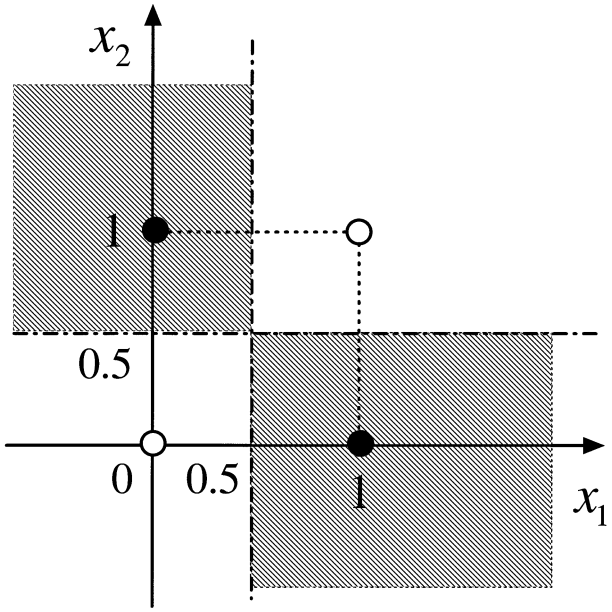

Fig. 9. Shaded area represents the on region for neuron N in Fig. 8. However, inputs are only from the set S.

TABLE IV
XOR NET PARAMETERS (ONE-CLASS), EX.5

D_k	w^1_{1k}	w^1_{2k}	w^0_{1k}	w^0_{2k}	p_k
D_1	0	0	-1	-1	$+1$
D_2	0	$+\infty$	$-\infty$	-1	-1
D_3	$+\infty$	0	-1	$-\infty$	-1

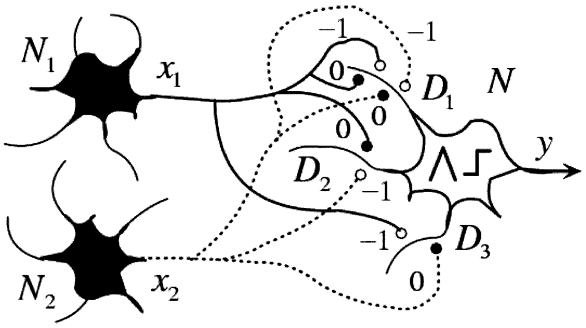

Fig. 10. Single morphological neuron solution to the XOR problem using three dendrites. The *on* region for the neuron N consists of the two points in $S = \{(0, 1), (1, 0)\}$, all other points of $\mathbb{R}^2$ are in the off region.

problem in pattern recognition; i.e., classify all points belonging to some region $X \subset \mathbb{R}^2$ as belonging to class C_1 and all other points as belonging to its complement (not in C_1) denoted by $C_0 = C_1'$. Embedding the XOR problem into $\mathbb{R}^2$—i.e, classifying all points in $\mathbb{R}^2$ as belonging to class C_1 if $\mathbf{x} \in \{(0, 1), (1, 0)\}$, to class C_2 if $\mathbf{x} \in \{(0, 0), (1, 1)\}$, and all remaining points as not belonging to either of these two classes—becomes a *two class* problem. As in classical perceptron theory, solving this problem requires two output neurons. However, in contrast to perceptron theory, no hidden layer is required for this special case of the two class problem.

Example 6: If M_1 and M_2 denote the two output neurons, then setting $M_1 = N$, where N denotes the neuron from the preceding example solves one part of the problem. The morphology and action of M_2 is similar to that of M_1 specifically, we want $f(\tau(\mathbf{x})) = 1$ if and only if $\mathbf{x} \in C_2 = \{(0, 0), (1, 1)\}$ and $f(\tau(\mathbf{x})) = 0$ whenever $\mathbf{x} \notin C_2$. Thus, if $\mathbf{y} = (y_1, y_2)$, where y_i denotes the output signal of M_i $(i = 1, 2)$, then the desired network output should be

$$\mathbf{y} = \begin{cases} (1, 0), & \mathbf{x} \in C_1 \\ (0, 1), & \mathbf{x} \in C_2 \\ (0, 0), & \mathbf{x} \in \mathbb{R}^2 - (C_1 \cup C_2). \end{cases}$$

Here, $\mathbf{y} = (f(\tau^1(\mathbf{x})), f(\tau^2(\mathbf{x})))$, $\tau^i(\mathbf{x}) = \bigwedge_{k=1}^{K_i} \tau_k^i(\mathbf{x})$ denotes the computation performed by M_i, and K_i denotes the number of dendrites of M_i. The morphological neural network shown in Fig. 11 solves this particular two-class problem. In this type of multiclass problem, w_{ijk}^l denotes the strength of the axonal branch of neuron N_i terminating on the kth dendrite of neuron M_j. The input response of this terminal branch is denoted by r_{ijk}^l. However, since this value is completely determined by l which is already specified by the superscript of the corresponding weight w_{ijk}^l we shall, henceforth, only specify the weights of the terminal fibers when presenting a network in tabular form. We will also use the notation D_{jk} and p_{jk} in order to denote the kth dendrite of M_j and its corresponding output response. For the specific two-class problem under consideration, Table V specifies the network illustrated in Fig. 11.

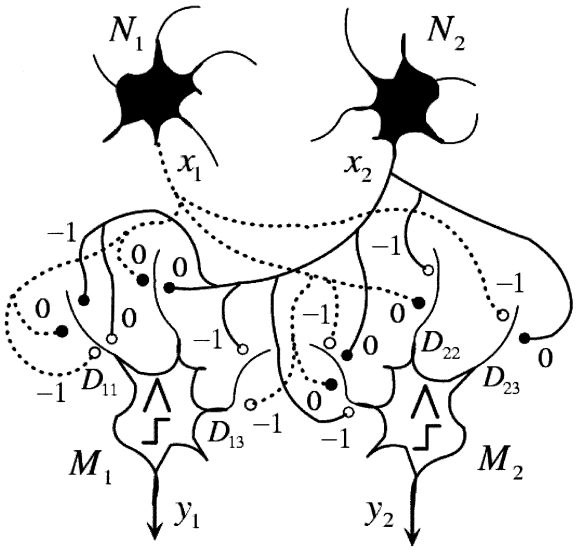

Fig. 11. Morphological neural network solves the XOR problem for points from the domain $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$.

TABLE V
XOR NET PARAMETERS (TWO-CLASS), EX.6

D_{jk}	w_{1jk}^1	w_{1jk}^0	w_{2jk}^1	w_{2jk}^0	p_{jk}
D_{11}	0	-1	0	-1	$+1$
D_{12}	0	$-\infty$	0	$-\infty$	-1
D_{13}	$+\infty$	-1	$+\infty$	-1	-1
D_{21}	0	-1	0	-1	$+1$
D_{22}	0	$-\infty$	$+\infty$	-1	-1
D_{23}	$+\infty$	-1	0	$-\infty$	-1

V. COMPUTATIONAL CAPABILITY OF A MORPHOLOGICAL NEURON WITH DENDRITIC STRUCTURE

Analogous to the classical single-layer perceptron (SLP) with one output neuron, a *single-layer morphological perceptron* (SLMP) with one output neuron also consists of a finite number of input neurons that are connected via axonal fibers to the output neuron. However, in contrast to an SLP, the output neuron of an SLMP has a dendritic structure and performs the lattice computation embodied by (10). Fig. 3 provides a pictorial representation of a general SLMP with a single output neuron. As the examples of the preceding section illustrate, the computational capability of an SLMP is vastly different from that of an SLP as well as that of classical perceptrons in general. No hidden layers were necessary to solve the XOR problem or to specify the points of the nonconvex region of Fig. 6. Observing differences by examples, however, does not provide answer as to the specific computational capabilities of an SLMP with one output neuron. Such an answer is given by the following theorem.

Theorem 1: If $X \subset \mathbb{R}^n$ is compact and $\varepsilon > 0$, then there exists a single layer morphological perceptron that assigns every

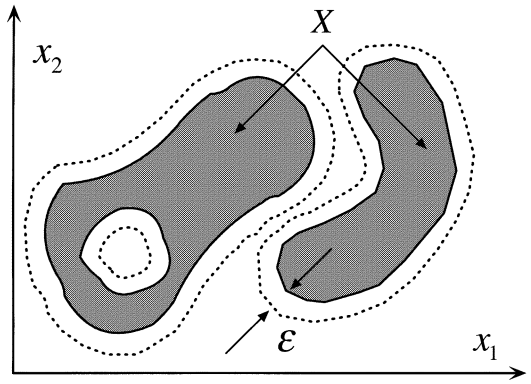

Fig. 12. Compact region X (shaded) and the banded region of thickness ε (dotted).

point of X to class C_1 and every point $\mathbf{x} \in \mathbb{R}^n$ to class C_0 whenever $d(\mathbf{x}, X) > \varepsilon$.

The expression $d(\mathbf{x}, X)$ in the theorem refers to the distance of the point $\mathbf{x} \in \mathbb{R}^n$ to the set X. Fig. 12 illustrates this concept. All points of X will be classified as belonging to class C_1 and all points outside the *banded* region of thickness ε will be classified as belonging to class C_0. Points within the banded region maybe misclassified. As a consequence, any compact configuration, whether it is convex or nonconvex, connected or not connected, contains a finite or infinite number of points, can be approximated to any desired degree of accuracy $\varepsilon > 0$ by an SLMP with one output neuron. The proof of the theorem requires tools from elementary point set topology and is given in the Appendix. Although the proof is an existence proof, part of it is constructive and provided the basic idea behind the training algorithm for SLMPs outlined in the next section.

VI. Dendrite Learning Algorithm

In this section we describe a learning procedure for dendrite computation in a single layer morphological perceptron. The set $T = \{(\mathbf{x}^\xi, c_\xi)\colon \xi = 1, \ldots, m\}$ denotes the training or learning set of exemplars $\mathbf{x}^\xi = (x_1^\xi, \ldots, x_n^\xi) \in \mathbb{R}^n$ and $c_\xi \in \{0, 1\}$ stands for the corresponding class number in a *one-class* problem, i.e., $c_\xi = 1$ if $\mathbf{x}^\xi \in C_1$, and $c_\xi = 0$ if $\mathbf{x}^\xi \in C_0 = C_1'$. From the formula given in (7) for single neuron computation based on lattice algebra, the response of the kth dendrite for input $\mathbf{x}^\xi$ is rewritten as follows:

$$\tau_k(\mathbf{x}^\xi) = p_k \bigwedge_{i \in I} \bigwedge_{l \in L} (-1)^{(1-l)}(x_i^\xi + w_{ik}^l). \tag{15}$$

For convenience, we repeat the meaning of the variables in (15) in order to make clear the steps involved in the training; $i \in I \subseteq \{1, \ldots, n\}$ denotes the fact that the axon of the ith input neuron, N_i, has terminal fibers that synapse on the kth dendrite of the single computing neuron N, $l \in L \subseteq \{0, 1\}$ denotes the fact that the axon of the i input neuron, N_i, has the r_{ik}^lth synapse $= (-1)^{(1-l)}$ on the kth dendrite of N. If there are no terminal fibers coming from the ith input neuron into the kth dendrite then $L = \emptyset$, if there is a single terminal fiber coming from the ith input neuron into the kth dendrite then $L = \{1\}$ (if excitatory) or $L = \{0\}$ (if inhibitory) but not both, or if there are two terminal fibers, one excitatory and one inhibitory, then $L = \{0, 1\}$. The activation function used is the Heaviside or unit step hard limiter defined in (9). A mathematical description of the training algorithm is given below:

*Algorithm 1: **SLMP** Training:*

Step 1) Initialize parameters and auxiliary index sets; also set weights of the hyperbox enclosing class C_1 (first dendrite). Let $k = 1, P = \{1, \ldots, m\}, I = \{1, \ldots, n\}$, $L = \{0, 1\}$, and $\forall\, i \in I$ compute

$$w_{ik}^1 = -\bigwedge_{c_\xi = 1} x_i^\xi; \qquad w_{ik}^0 = -\bigvee_{c_\xi = 1} x_i^\xi.$$

Step 2) Compute the response of the current dendrite.

$$p_k = (-1)^{\operatorname{sgn}(k-1)}$$
$$r_{ik}^l = (-1)^{(1-l)}; \qquad \forall\, i \in I,\ l \in L$$
$$\tau_k(\mathbf{x}^\xi) = p_k \bigwedge_{i \in I} \bigwedge_{l \in L} r_{ik}^l (x_i^\xi + w_{ik}^l), \qquad \forall\, \xi \in P.$$

Note: $\operatorname{sgn}(x)$ denotes the signum function.

Step 3) Compute the total response of the output neuron N.

$$\tau(\mathbf{x}^\xi) = \bigwedge_{j=1}^{k} \tau - j(\mathbf{x}^\xi), \qquad \forall\, \xi \in P.$$

Step 4) Test if with the generated k dendrites learning is successful using the given hard limiter.

$$\forall\, \xi \in P, \qquad \text{does } f(\tau(\mathbf{x}^\xi)) = c_\xi?$$

If true, output the final weights and the input-output synaptic responses of the completed network; K denotes the final number of dendrites grown in neuron N upon convergence. Let $K = k$. For $k = 1, \ldots, K$ and $\forall\, i \in I$, $l \in L$ print network parameters (the algorithm ends here)

$$k,\ w_{ik}^l,\ r_{ik}^l,\ p_k \qquad \text{STOP}.$$

If false, training continues (next step) by growing additional dendrites.

Step 5) Add a new dendrite to N and initialize several auxiliary and index sets; initially, set D gets all class 1 patterns. Let $k = k + 1$, $I = I' = X = E = H = \emptyset$, and $D = C_1$

Step 6) Select a misclassified pattern from class C_0.

$$\text{choice}(\mathbf{x}^\gamma) \in C_0 \ni f(\tau(\mathbf{x}^\gamma)) = 1$$

Note: γ above is the index of the misclassified pattern in C_0, and function choice(x) is a proper selection mechanism (e.g., random or sort selection).

Step 7) Compute the minimum *Chebyshev* or *chessboard* distance from the selected misclassified pattern $\mathbf{x}^\gamma$ to all patterns in set D.

$$\mu = \bigwedge_{\xi \neq \gamma} \left\{ \bigvee_{i=1}^{n} |x_i^\gamma - x_i^\xi| \colon \mathbf{x}^\xi \in D \right\}.$$

Note: since set D is updated in Step 12, the minimum distance μ changes during the generation of new terminals on the same dendrite.

Step 8) Keep indexes and coordinates of patterns in set D that are on the border of the hyperbox centered at $\mathbf{x}^\gamma$.

$$I' = \{i\colon |x_i^\gamma - x_i^\xi| = \mu,\ \mathbf{x}^\xi \in D,\ \xi \neq \gamma\}$$
$$X = \{(i, x_i^\xi)\colon |x_i^\gamma - x_i^\xi| = \mu,\ \mathbf{x}^\xi \in D,\ \xi \neq \gamma\}.$$

Step 9) Assign weights and input values for new axonal fibers in the current dendrite that provide correct classification for the misclassified pattern.

$\forall (i, x_i^\xi) \in X$

(a) if $x_i^\xi < x_i^\gamma$ set $w_{ik}^1 = -x_i^\xi$, $E = \{1\}$

(b) if $x_i^\xi > x_i^\gamma$ set $w_{ik}^0 = -x_i^\xi$, $H = \{0\}$

Step 10) Update index sets I and L (only those input neurons and fibers needed to classify $\mathbf{x}^\gamma$ correctly).

$$I = I \cup I'; \qquad L = E \cup H$$

Step 11) Keep C_1 exemplars $\mathbf{x}^\xi (\xi \neq \gamma)$ that do not belong to the recently created region in Step 9.

$$D' = \{\mathbf{x}^\xi \in D\colon \forall\, i \in I,\ -w_{ik}^1 < x_i^\xi \text{ and } x_i^\xi < -w_{ik}^0\}$$

Note: auxiliary set D' is used for updating set D that is reduced or shrunk during the creation of new possible axonal fibers on the current dendrite.

Step 12) Check if there is no need to wire more axonal fibers to the current dendrite.

If $D' = \emptyset$ then return to Step 2 else set $D = D'$ and loop to Step 7.

Note: Going back to Step 2 means that the current dendrite is done, no more terminal fibers need to be wired in it but the neuron response must be recomputed to see if learning has been accomplished; returning to Step 7 means that the current dendrite needs additional fibers.

The following examples provide some applications of Algorithm 1 that fall under the category of one class problems previously mentioned in Section V. Thus, the learning procedure of the SLMP demonstrates its computation capability for solving *nonlinear* problems in a simple and direct way by generating the necessary neuron dendritic structure without convergence problems or need of hidden layers. It is also of interest to note that the training algorithm grows new axonal fibers as it learns to recognize complex pattern shapes. This is in agreement with the growth of neural fibers and dendritic connections during learning in the cerebral cortex.

Example 7: Let $X \subset \mathbb{R}^3$ be the set $\{(x_1, x_2, x_3)\colon x_i \in \{0, 1\}, i = 1, 2, 3\}$ which consists of all binary triples, i.e., $X = \{(0, 0, 0), (0, 0, 1), \ldots, (1, 1, 1)\}$. The class of pattern $\mathbf{x}^\xi$ for $\xi = 1, \ldots, 8$ is defined as the exclusive or of its coordinates, $c_\xi = x_1^\xi \oplus x_2^\xi \oplus x_3^\xi$ for all ξ. Fig. 13 shows the corresponding learning set $T = \{(\mathbf{x}^\xi, c_\xi)\colon \mathbf{x}^\xi \in X, c_\xi \in \{0, 1\}\}$ for the three-dimensional XOR recognition problem; a solid dot ($\bullet$) represents a point in class C_1 ($c_\xi = 1$), an open dot represents a point in class C_0 ($c_\xi = 0$). If the set X is embedded in $\mathbb{R}^3$, all points $\mathbf{x}$ belonging to its complement are assigned to class C_0 (*rejection* class). The data in the set X are sorted in ascending order using the numerical expression $4x_1 + 2x_2 + x_3 + 1$ which is the decimal equivalent of the binary vector plus one. Hence, $(0, 0, 0)$ is the first point and $(1, 1, 1)$ is the eighth point. The rejected patterns $\mathbf{x}^\gamma$ occur for $\gamma = 1, 4, 6, 7$ and each one of them is removed in that order by Algorithm 1. Table VI gives the weights and output responses for the dendritic structure that solves the problem; recall that $r_{ik}^1 = +1$ and $r_{ik}^0 = -1$ are the corresponding input responses for $i \in \{1, 2, 3\}$ and $k \in \{1, \ldots, 5\}$. The 3-D XOR problem is equivalent to the *n-parity problem* with $n = 3$ and it has been pointed out, e.g., see [48], [47], that dynamic node creation algorithms for feedforward networks need two or three units in a single hidden layer to classify correctly all the inputs from X.

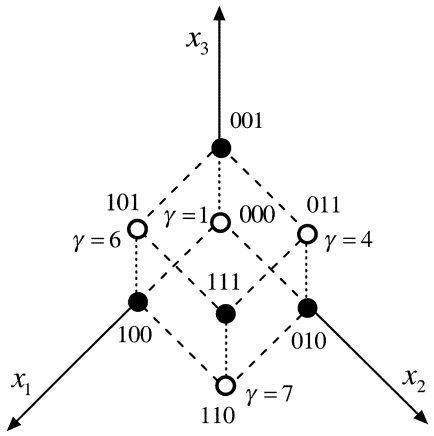

Fig. 13. Discrete set X for the 3-D XOR problem. Patterns of class C_0 are eliminated in ascending order of γ.

TABLE VI
WEIGHTS AND OUTPUT RESPONSES OF THE MORPHOLOGICAL NEURON THAT SOLVES THE 3-D XOR PROBLEM

D_k	w_{1k}^1	w_{2k}^1	w_{3k}^1	w_{1k}^0	w_{2k}^0	w_{3k}^0	p_k
1	0	0	0	-1	-1	-1	$+1$
2	∞	∞	∞	-1	-1	-1	-1
3	∞	0	0	-1	$-\infty$	$-\infty$	-1
4	0	∞	0	$-\infty$	-1	$-\infty$	-1
5	0	0	∞	$-\infty$	$-\infty$	-1	-1

Example 8: Fig. 14 displays a scatter diagram in two dimensions of a training set T used as input to the SLMP algorithm. This set corresponds to a sample projection in x_1 and x_2 obtained from a 3-D discrete data set related to a practical one-class problem concerned with the discrimination between mines and false alarms (abbreviated as FA) [19], [20]. In the given figure, mines are represented by solid dots ($\bullet$) and false alarms are shown as small circles ($\circ$). Specifically, the set T has 81 patterns of which 39 are false alarms and 42 are mines; Table VII gives the net parameters for the morphological neuron that classifies all the points correctly. Fig. 14 also illustrates the *step* regions (tunnels or boxes) associated with each dendrite. Dendrite D_1 encloses all mines and rejects 34 FA; dendrites D_2, D_3, D_4 to D_5 remove, respectively, the false alarms inside

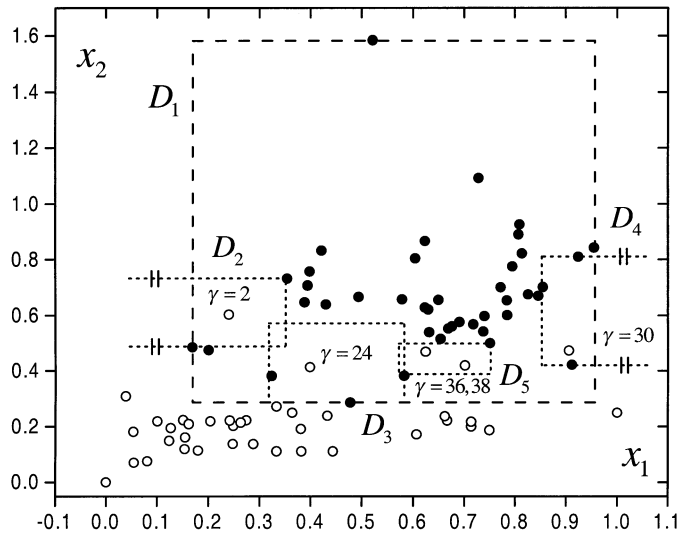

Fig. 14. Set T for the mines ($\bullet$) and false alarms ($\circ$) problem. The dotted lines mark the step regions generated by each dendrite D_k for $k = 1, \ldots, 5$.

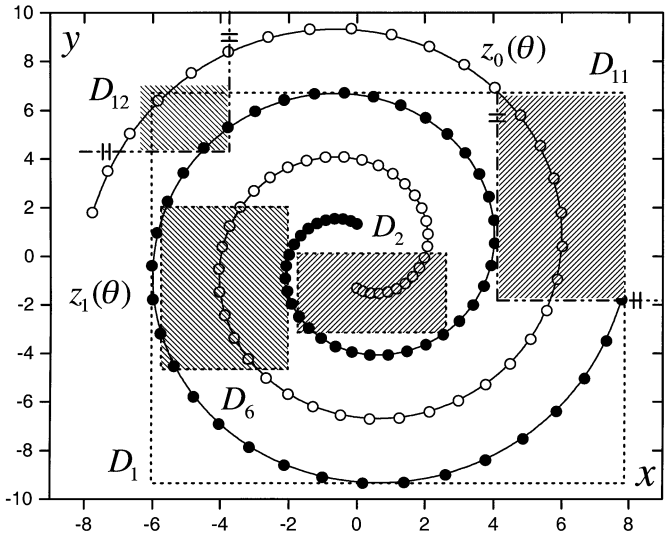

Fig. 15. Set T_1 for the two spiral problem. Points of spiral z_1 are shown as solid dots, points of spiral z_0 are shown as circles. Example local regions produced by dendrites appear shaded except the box enclosing points of class C_1.

TABLE VII
PARAMETER VALUES FOR THE MORPHOLOGICAL NEURON OF EX.8

D_k	w^1_{1k}	w^1_{2k}	w^0_{1k}	w^0_{2k}	p_k
1	-0.169	-0.287	-0.955	-1.584	$+1$
2	$+\infty$	-0.485	-0.388	-0.732	-1
3	-0.324	-0.287	-0.583	-0.598	-1
4	-0.855	-0.422	$-\infty$	-0.810	-1
5	-0.579	-0.383	-0.751	-0.515	-1

TABLE VIII
PARAMETER VALUES FOR THE MORPHOLOGICAL NEURON OF EX.9

D_k	w^1_{1k}	w^1_{2k}	w^0_{1k}	w^0_{2k}	p_k
1	6.22	9.35	-7.77	-6.71	$+1$
2	1.70	2.97	-3.01	-0.85	-1
6	5.77	4.53	2.09	-2.25	-1
11	-4.04	1.80	$-\infty$	$-\infty$	-1
12	$+\infty$	-4.44	3.80	$-\infty$	-1

the box whose index value γ is 2,24,30 and $\{36, 38\}$. The input responses, in this case, are given by $r^1_{ik} = +1$ and $r^0_{ik} = -1$ for $i \in \{1, 2\},\ k \in \{1, \ldots, 5\}$.

Example 9: A nontrivial benchmark for testing the performance of a training algorithm is the well known problem of the *two intertwined spirals* [25]. Several researchers have provided solutions to this problem using different artificial network architectures with several hidden layers [25], interconnected hidden units [16], a single hidden layer for small training sets [42], or a dynamic node creation mechanism with a number of hidden units ranging from 28 to 38 for a training set with less than 1000 points for the two spirals [46], [48].

For this example, we use two *Archimedean* spirals defined by the complex expression $(i = \sqrt{-1})$

$$z_c(\theta) = x_c(\theta) + i\alpha^{-1} y_c(\theta) = (-1)^{1-c}\rho\theta e^{i\theta} \qquad (16)$$

where $c \in \{0, 1\}$ denotes the spiral class label, θ is the angle in radians, $\alpha > 0$ is the aspect ratio between the x_c and y_c coordinates (in a mathematical context, $\alpha = 1$, but it is common practice for computer generated curves to take $\alpha > 0$), and $\rho > 0$ is a constant that determines the spread of the spiral turns. In order that the initial points of both spirals are $(x_0, y_0) = (0, -\alpha)$ and $(x_1, y_1) = (0, +\alpha)$, respectively, the value of ρ was set to $2/\pi$ and θ was sampled in the interval $[\pi/2, 4\pi)$; also, for on screen display we set $\alpha = 4/3$. From (16), the corresponding parametric expressions that were used to generate the training set T_1 are given by

$$x_c(\theta) = \frac{2(-1)^{1-c}}{\pi}\,\theta\cos\theta \qquad (17)$$

$$y_c(\theta) = \frac{8(-1)^{1-c}}{3\pi}\,\theta\sin\theta. \qquad (18)$$

Notice that, $z_0(\theta) + z_1(\theta) = 0$ for all values of θ, hence $x_0(\theta) = -x_1(\theta)$ and $y_0(\theta) = -y_1(\theta)$. Consequently, if the point (x, y) belongs to spiral z_1 then the *symmetric* point $(-x, -y)$ belongs to the other spiral, i.e., z_0. It is important to remark that this *a priori* knowledge regarding the nature of the set under discussion shall not be used in a generic learning algorithm since it will bias its intrinsic capability for generalization. However, in order to optimize and adjust the performance of a training mechanism used for solving real-world problems this constraint is usually relaxed. Algorithm1 for the SLMP training *does not* use any *a priori* knowledge about set T_1 (same assumption for he two preceding examples).

Fig. 15 shows the two continuous spirals (solid curves), the 128 points in set T_1 (64 points in each spiral), and some example step regions (boxes or corners) associated with the corresponding dendrites grown through the learning stage. To solve

TABLE IX
PERFORMANCE RESULTS OF THE SLMP TRAINING ALGORITHM FOR SEVERAL SAMPLED SETS IN THE TWO SPIRAL PROBLEM

T_m	2^{6+m}	h	t_T (s)	K
T_1	128	0.172	0.49	12
T_2	256	0.086	0.68	13
T_3	512	0.043	1.82	16
T_4	1024	0.022	3.24	15
T_5	2048	0.011	6.41	15
T_6	4096	0.005	12.86	15
T_7	8192	0.003	25.85	15

the problem, the morphological neuron required $K = 12$ dendrites; Table VIII lists the weights and output responses only for dendrites D_1, D_2, D_6, D_{11}, and D_{12} whose action is depicted in Fig. 15. Although the numerical values are displayed with three digits, the computations were performed using 11 significant figures.

A second experiment was conducted to see if an increase in the number of points in T_1 would require more dendrites to solve the problem. Six additional training sets, $T_2, \ldots, T_7$ were generated, each with a number of points given by 2^{6+m} for $m = 1, \ldots, 7$. Table IX shows the results obtained with complete classification in each training set; the value of h in the third column is the sample step size of $\theta \in [\pi/2, 4\pi)$, and the training time t_T in seconds is given in the fourth column. It can be observed that the number of dendrites $(K = 15)$ stabilizes after T_4 (1024 points) which compares favorably to other techniques, as described earlier, that use optimized or improved backpropagation algorithms. This example clearly demonstrates that dendrite computation in single neurons do not require hidden layers, there are no convergence problems, learning is achieved rapidly employing simple arithmetical operations, and the number of dendrites and synapses is fairly small (e.g., when compared with the typical synaptic packing density in biological neurons [24], [43]).

VII. CONCLUSION

We presented a new paradigm for single morphological neuron computation. This paradigm takes into account inhibitory and excitatory input and output responses as well as computation in dendrites. A training algorithm was presented that grows new axonal fibers and dendritic structures during the learning phase. In this sense, the morphological model mimics the biological model more closely than the neurons used in traditional artificial neural networks. Theorem 1 as well as the examples presented make it obvious that a SLMP with one output neuron is far more powerful than a SLP with one output neuron or a perceptron with one output neuron and one hidden layer.

Questions may be raised as to whether dendrites merely represent hidden layers in disguise. Such questions are valid in light of the fact that, theoretically, a two hidden layer perceptron can also classify any compact region in $\mathbb{R}^n$. However, there are some major differences between the model presented here, and hidden layer perceptrons. In comparison to hidden layer neurons which generally use sigmoidal activation functions, dendrites have no activation functions. They only compute the basic logic functions of AND, OR, and NOT. Activation takes place only within the neuron via the hard-limiter function. Also, with hidden layers the number of neurons within a hidden layer is predetermined before training of weights, which traditionally involves backpropagation methods. In our model, dendrites are grown automatically as the neuron learns its specific task. Furthermore, *no* error remains after training. *All* pattern vectors of the training set will *always* be correctly identified after the training stops.

Although the topology of the neural model presented in this paper seems more complex than that of other models used in traditional artificial neural networks, it remains only a rough approximation of the extremely complex biological neuron. A pertinent observation concerning the proposed model is that an input neuron is allowed to have both excitatory and inhibitory terminal axon fibers. From a biological perspective, this is incorrect. One way around this problem is to send all input values to an intermediate inhibitory neuron first. This inhibitory neuron then sends only inhibitory signals to N. The terminal fibers of the inhibitory neuron will have weight values identical to the weight values of the inhibitory fibers of the original input neuron while the input neuron will only have its original excitatory input response at the terminals on N. However, as far as we have been able to determine thus far, this complicates the neural network and increases the computational overhead without additional benefit to the computational abilities of our model.

Another relevant observation concerns the training algorithm. Although Algorithm 1 is fairly straightforward and fast convergence guaranteed for most finite training sets, *a priori* knowledge regarding the nature of the patterns under consideration was not used in the examples in order to preserve the algorithm's intrinsic capability for generalization. Thus, if a pattern $\mathbf{x}$ belonging to class C_1 lies on the boundary of the final configuration determined by the training algorithm, then any other C_1 pattern arbitrarily close to $\mathbf{x}$ but *outside* the configuration will be misclassified as a C_0 pattern. This can be problematic in real applications. However, in real-world problems some *a priori* knowledge of pattern feature behavior or expectation is often available. For instance, maximum and minimum GPR (ground penetrating radar) responses for certain types of buried metal mines have been well established. Similarly, the Chebyshev distance between the training patterns belonging to class C_1 and C_0 can be easily computed. These kinds of information can be used to make the boundary pieces associated with the weights more pliable by using fuzzy boundary control parameters α_{ik} and β_{ik}. These parameters are appropriately added (or substracted) to the weights in Step 1 and Step 9 during training. As a final observation, in comparison to perceptrons, training sets will always be 100% correctly classified.

The lattice algebra approach and dendritic computing in ANNs is extremely new and a multitude of open questions await further exploration. For example, the two output neurons of Example 6 indicates an obvious approach to handle multiclass discrimination. The base of applications needs to be expanded and compared to traditional methods. A rigorous mathematical comparison between the proposed model and other traditional

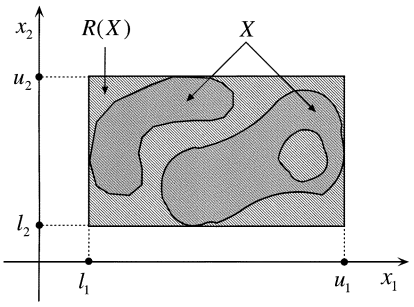

Fig. 16. Compact subset X is indicated by the shaded region while the smallest rectangle $R(X)$ enclosing X is hatched.

models needs to be established. We hope that these problems will generate sufficient interest and entice other researchers to join our efforts in advancing the frontiers of this new endeavor.

APPENDIX

The distance function used in the proof of Theorem 1 is the *Chebyshev* or *chessboard* distance. Specifically, if $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, then the Chebyshev distance between $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$d(\mathbf{x}, \mathbf{y}) = \max\{|x_i - y_i|: 1 \le i \le n\}. \tag{19}$$

Given $\mathbf{x} \in \mathbb{R}^n$ and $X \subset \mathbb{R}^n$, the distance from $\mathbf{x}$ to X in terms of the Chebyshev distance is given by

$$d(\mathbf{x}, X) = \min\{d(\mathbf{x}, \mathbf{y}): \mathbf{y} \in X\}. \tag{20}$$

We need to point out that Theorem 1 also holds for the Euclidean distance. A simple replacement of ε by $\varepsilon' = \varepsilon/\sqrt{2}$ in the proof below will make the theorem hold true for the Euclidean distance.

Theorem 1, Section V:

Proof: For $i \in \{1, \ldots, n\}$ let N_i denote the ith input neuron, p_i: $X \to \mathbb{R}$ the ith projection defined by $p_i(\mathbf{x}) = x_i$, and let N denote the output neuron. Define

$$\begin{aligned} l_i &= \inf\{p_i(\mathbf{x}): \mathbf{x} \in X\} \\ u_i &= \sup\{p_i(\mathbf{x}): \mathbf{x} \in X\}. \end{aligned} \tag{21}$$

Since X is a compact set and each p_i is continuous on X, the numbers given by l_i and u_i exists for every i. Let $R(X)$ denote the smallest hyperbox containing X; i.e.,

$$R(X) = \{\mathbf{x} \in \mathbb{R}^n: l_i \le x_i \le u_i,\ i = 1, \ldots, n\}. \tag{22}$$

Fig. 16 illustrates the two-dimensional case. There are two relationships between X and $R(X)$ that need to be considered. The trivial case arises when $X = R(X)$. In this case, let N have one dendrite D_1 with output response $p_1 = 1$ and define the weights of the axonal branches of N_i terminating on this dendrite by

$$w_{i1}^1 = -l_i \quad \text{and} \quad w_{i1}^0 = -u_i. \tag{23}$$

The SLMP thus defined satisfies the conclusion of the theorem. In order to verify this claim, note that for every $\mathbf{x} \in X = R(X)$, the ith coordinate of $\mathbf{x}$ obeys the following inequalities:

$$\begin{aligned} &l_i \le x_i \le u_i \\ &0 \le x_i - l_i = x_i + w_{i1}^1 \\ &0 \le u_i - x_i = -(x_i + w_{i1}^0). \end{aligned} \tag{24}$$

Therefore

$$\tau(\mathbf{x}) = \tau_1(\mathbf{x}) = \bigwedge_{i=1}^{n} \bigwedge_{l=0}^{1} (-1)^{1-l}(x_i + w_{i1}^l) \ge 0 \tag{25}$$

and $f(\tau(\mathbf{x})) = 1$ for all $\mathbf{x} \in X$. If $\mathbf{x} \in \mathbb{R}^n$ with $d(\mathbf{x}, X) > \varepsilon$ then $\mathbf{x} \notin X = R(X)$. It follows that there exists an index $i \in \{1, \ldots, n\}$ such that $x_i < l_i$ or $x_i > u_i$. Hence, either

$$x_i - l_i = x_i + w_{i1}^1 < 0$$

or

$$-(x_i - u_i) = -(x_i + w_{i1}^0) < 0. \tag{26}$$

As a consequence, $\tau(\mathbf{x}) = \tau_1(\mathbf{x}) < 0$ and, therefore, $f(\tau(\mathbf{x})) = 0\ \forall \mathbf{x} \in \mathbb{R}^n$ for which $d(\mathbf{x}, X) > 0$. In fact, it is obvious that $f(\tau(\mathbf{x})) = 0\ \forall \mathbf{x} \notin X$ even if $d(\mathbf{x}, X) < \varepsilon$. This proves the theorem when $R(X) = X$.

If $X \ne R(X)$, set $Y = R(X) - X$ and let $\overline{Y}$ be the closure of Y in $\mathbb{R}^n$. Since $\overline{Y} \subseteq R(X)$, $\overline{Y}$ is both closed and bounded, therefore, also compact. Define for each $\mathbf{y} \in \overline{Y}$ the neighborhood

$$U(\mathbf{y}) = \left\{\mathbf{x} \in \mathbb{R}^n: d(\mathbf{x}, \mathbf{y}) < \frac{\varepsilon}{2}\right\}. \tag{27}$$

The collection $\mathcal{U} = \{U(\mathbf{y}): \mathbf{y} \in \overline{Y}\}$ is an open cover for $\overline{Y}$ and since $\overline{Y}$ is compact, there is a finite subcollection of $\mathcal{U}$ which also covers $\overline{Y}$. Let $\mathcal{U}_m = \{U(\mathbf{y}^k): k = 1, \ldots, m\}$ denote this finite subcollection and for each k define

$$\overline{U}(\mathbf{y}^k) = \left\{\mathbf{x} \in \mathbb{R}^n: d(\mathbf{x}, \mathbf{y}^k) \le \frac{\varepsilon}{2}\right\} \tag{28}$$

i.e., $\overline{U}(\mathbf{y}^k)$ is the closure of $U(\mathbf{y}^k)$ in $\mathbb{R}^n$. There are two specific cases that occur in relationship to $\mathcal{U}_m$, X, and ε.

Case 1. $U(\mathbf{y}^k) \cap X \ne \emptyset\ \forall k \in \{1, \ldots, m\}$. This, again, turns out to be a trivial case as the SLMP defined earlier in the proof—(23)—satisfies the conclusion of the theorem. Obviously, if $\mathbf{x} \in X$, then the ith coordinate of $\mathbf{x}$ still satisfies the inequalities of (24) for all $i = 1, \ldots, n$. Next suppose $\mathbf{x} \in \mathbb{R}^n$ with $d(\mathbf{x}, X) > \varepsilon$, then $\mathbf{x} \notin R(X)$. For if $\mathbf{x} \in R(X)$, then $\mathbf{x} \in Y = R(X) - X$. Hence there must exist a member $U(\mathbf{y}^k) \in \mathcal{U}_m$ such that $\mathbf{x} \in U(\mathbf{y}^k)$. Since by Case 1 assumption $U(\mathbf{y}^k) \cap X \ne \emptyset$, there exists a point $\mathbf{z} \in X$ and $\mathbf{z} \in U(\mathbf{y}^k)$. But then, $d(\mathbf{x}, X) \le d(\mathbf{x}, \mathbf{z}) \le d(\mathbf{x}, \mathbf{y}^k) + d(\mathbf{y}^k, \mathbf{z}) < (\varepsilon/2) + (\varepsilon/2) = \varepsilon$ which contradicts the fact that $d(\mathbf{x}, X) > \varepsilon$. It follows that if $d(\mathbf{x}, X) > \varepsilon$, then $\mathbf{x} \notin R(X)$. Hence, for each such $\mathbf{x}$ there must exist an index $i \in \{1, \ldots, n\}$ such that the ith coordinate of $\mathbf{x}$ satisfies one of the inequalities of (26), so that $f(\tau(\mathbf{x})) = 0$. This proves the theorem if Case 1 occurs.

Case 2. Suppose Case 1 does not occur. In this case some elements of $\mathcal{U}_m$ do not intersect X. Let $\{U(\mathbf{y}^{s_k}): k = 1, \ldots, K-1\}$ denote the maximal subcollection of $\mathcal{U}_m$ for which $U(\mathbf{y}^{s_k}) \cap$

$X = \emptyset$. For each $i \in \{1, \ldots, n\}$ and each $k \in \{1, \ldots, K-1\}$ define the bounds

$$l_{ik} = y_i^{s_k} - \frac{\varepsilon}{2} = \inf\{p_i(\mathbf{x}): \mathbf{x} \in U(\mathbf{y}^{s_k})\}$$
$$u_{ik} = y_i^{s_k} + \frac{\varepsilon}{2} = \sup\{p_i(\mathbf{x}): \mathbf{x} \in U(\mathbf{y}^{s_k})\}. \quad (29)$$

As before, let N_i denote the input neurons. However, the output neuron has K dendrites $D_1, \ldots, D_K$, where the output responses are given by

$$p_k = \begin{cases} +1 \Leftrightarrow k = 1 \\ -1 \Leftrightarrow k = 2, \ldots, K. \end{cases} \quad (30)$$

The weights of the axonal branches of N_i terminating in the kth dendrite of N are defined by

$$w_{ik}^0 = \begin{cases} -u_i \Leftrightarrow k = 1 \\ -u_{ik-1} \Leftrightarrow k = 2, \ldots, K \end{cases} \quad (31)$$

and

$$w_{ik}^1 = \begin{cases} -l_i \Leftrightarrow k = 1 \\ -l_{ik-1} \Leftrightarrow k = 2, \ldots, K. \end{cases} \quad (32)$$

Here u_i and l_i are defined by (21) while u_{ik-1} and l_{ik-1} are defined by (29). We now show that the morphological perceptron thus defined satisfies the conclusion of the theorem. Suppose $\mathbf{x} \in X$. Since $U(\mathbf{y}^{s_k}) \cap X = \emptyset$, then $\mathbf{x} \notin U(\mathbf{y}^{s_k})$ for $k \in \{1, \ldots, K-1\}$. Thus, for any given k, there must exist an index $i \in \{1, \ldots, n\}$ (depending on k) such that either $x_i < l_{ik}$ or $u_{ik} < x_i$. Hence, either

$$x_i - l_{ik} = x_i + w_{ik+1}^1 < 0$$

or

$$-(x_i - u_{ik}) = -(x_i + w_{ik+1}^0) < 0. \quad (33)$$

Equation (33) implies that

$$\bigwedge_{i=1}^{n} \bigwedge_{l=0}^{1} (-1)^{1-l}(x_i + w_{ik+1}^l) < 0. \quad (34)$$

But $p_{k+1} = -1$ and, hence, $\tau_{k+1}(\mathbf{x}) > 0$. Since (34) holds for any given $k \in \{1, \ldots, K-1\}$ and (33) holds for any given $\mathbf{x} \in X$, we have that

$$\tau(\mathbf{x}) = \bigwedge_{k=1}^{K} \tau_k(\mathbf{x}) \geq 0, \qquad \forall \mathbf{x} \in X. \quad (35)$$

Therefore, $f(\tau(\mathbf{x})) = 1$ for all $\mathbf{x} \in X$.

Now suppose that $d(\mathbf{x}, X) > \varepsilon$. Then two cases might occur, namely either $\mathbf{x} \notin R(X)$ or $\mathbf{x} \in R(X)$. If $\mathbf{x} \notin R(X)$, then, as before, there must exist an index $i \in \{1, \ldots, n\}$ such that the ith coordinate of $\mathbf{x}$ satisfies one of the inequalities of Eq. (33). As a result, $\tau_1(\mathbf{x}) < 0$. Hence, $\tau(\mathbf{x}) < 0$ and $f(\tau(\mathbf{x})) = 0$. If $d(\mathbf{x}, X) > \varepsilon$ and $\mathbf{x} \in R(X)$, then since $\mathbf{x} \notin X$, $\mathbf{x} \in Y = R(X) - X$. But this means that $\mathbf{x} \in U(y^j)$ for some $j \in \{1, \ldots, m\}$. If $U(\mathbf{y}^j) \cap X \neq \emptyset$ then there exists a point $\mathbf{z} \in X$ and $\mathbf{z} \in U(\mathbf{y}^j)$. But then $d(\mathbf{x}, X) \leq d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}^k) + d(\mathbf{y}^k, \mathbf{z}) < (\varepsilon/2) + (\varepsilon/2) = \varepsilon$ contradicting the fact that $d(\mathbf{x}, X) > \varepsilon$. Therefore, $U(\mathbf{y}^j) \cap X = \emptyset$. Consequently, $j = s_k$ for some $k \in \{1, \ldots, K-1\}$. Since $\mathbf{x} \in U(\mathbf{y}^{s_k})$, $l_{ik} < x_i < u_{ik}$ for all $i \in \{1, \ldots, n\}$. Equivalently, we now have

$$0 < x_i - l_{ik} = x_i + w_{ik+1}^1$$
$$0 < u_{ik} - x_i = -(x_i + w_{ik+1}^0). \quad (36)$$

Hence

$$\bigwedge_{i=1}^{n} \bigwedge_{l=0}^{1} (-1)^{1-l}(x_i + w_{ik+1}^l) > 0. \quad (37)$$

But $p_{k+1} = -1$ and, therefore $\tau_{k+1}(\mathbf{x}) < 0$. It now follows that $\tau(\mathbf{x}) = \bigwedge_{k=1}^{K} \tau_k(\mathbf{x}) < 0$ and $f(\tau(\mathbf{x})) = 0$. Hence, $f(\tau(\mathbf{x})) = 0$ whenever $d(\mathbf{x}, X) > \varepsilon$. ■

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their valuable criticisms and helpful comments.

REFERENCES

[1] M. A. Arbib, Ed., *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1998.
[2] R. C. Backhouse and B. Carré, "Regular algebra applied to path-finding problems," *J. Inst. Math. Applicat.*, vol. 15, pp. 161–186, 1975.
[3] C. Benzaken, "Structures et algébre des cheminements," in *Network and Switching Theory*, G. Biorci, Ed. New York: Academic, 1968, pp. 40–57.
[4] B. Carré, "An algebra for network routing problems," *J. Inst. Math. Applicat.*, vol. 7, pp. 273–294, 1971.
[5] R. Cuninghame-Green, "Process synchronization in steel-works—A problem of feasibility," in *Proc. 2nd Int. Conf. Oper. Res.*, Banbury and Maitland, Eds: English Univ. Press, 1960, pp. 323–328.
[6] ——, "Describing industrial processes with interference and approximating their steady-state behavior," *Oper. Res.*, vol. 13, pp. 95–100, 1962.
[7] ——, *Minimax Algebra: Lecture Notes in Economics and Mathematical Systems 166*. New York: Springer-Verlag, 1979.
[8] J. L. Davidson, "Lattice Structures in the Image Algebra and Applications to Image Processing," Ph.D. dissertation, Univ. Florida, Gainesville, 1989.
[9] J. L. Davidson and G. X. Ritter, "A theory of morphological neural networks," *Proc. SPIE*, vol. 1215, pp. 378–388, July 1990.
[10] J. L. Davidson, "Simulated annealing and morphological neural networks," *Proc. SPIE*, vol. 1769, pp. 119–127, July 1992.
[11] ——, "Foundations and applications of lattice transforms in image processing," in *Advances in Electronics and Electron Physics*, P. Hawkes, Ed. Boston, MA: Academic, 1992, vol. 84, pp. 61–130.
[12] J. L. Davidson and F. Hummer, "Morphology neural networks: An introduction with applications," *IEEE Trans. Signal Processing*, vol. 12, pp. 177–210, Feb. 1993.
[13] J. L. Davidson and R. Srivastava, "Fuzzy image algebra neural network for template identification," in *Proc. 2nd Annu. Midwest Electro-Technology Conf.*, Ames, IA, Apr. 1993, pp. 68–71.
[14] J. L. Davidson and A. Talukder, "Template identification using simulated annealing in morphology neural networks," in *Proc. 2nd Annu. Midwest Electro-Technology Conf.*, Ames, IA, Apr. 1993, pp. 64–67.
[15] J. C. Eccles, *The Understanding of the Brain*. New York: McGraw-Hill, 1977.
[16] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Neural Information Processing Systems*, D. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 524–532.
[17] B. Giffler, "Mathematical solution of production planning and scheduling problems," IBM ASDD, Tech. Rep., 1960.
[18] H. J. A. M. Heijmans, *Morphological Image Operators*. Boston, MA: Academic, 1994.
[19] A. K. Hocaoglu and P. D. Gader, "Continuous processing of acoustic data for landmine detection," in *Proc. SPIE 19th AIPR Workshop Acoustics*, Orlando, FL, Apr. 2002.

[20] A. K. Hocaoglu, P. Gader, J. Keller, and B. Nelson, "Anti-personnel land mine detection and discrimination using acoustic data," Subsurface Sensing Technologies and Applications, 2001, to be published.
[21] W. R. Holmes and W. Rall, "Electronic models of neuron dendrites and single neuron computation," in *Single Neuron Comput.*, T. McKenna, J. Davis, and S. F. Zornetzer, Eds. New York: Academic, 1992, pp. 7–25.
[22] C. Koch and I. Segev, Eds., *Methods in Neuronal Modeling: From Synapses to Networks*. Cambridge, MA: MIT Press, 1989.
[23] C. Koch and T. Poggio, "Multiplying with synapses and neurons," in *Single Neuron Comput.*, T. McKenna, J. Davis, and S. F. Zornetzer, Eds. New York: Academic, 1992, pp. 315–345.
[24] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons*. New York: Oxford Univ. Press, 1999.
[25] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," in *Proc. 1988 Connectionist Model Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. San Mateo, CA: Morgan Kaufmann, 1988, pp. 52–59.
[26] T. McKenna, J. Davis, and S. E. Zornetzer, *Single Neuron Computation*. San Diego, CA: Academic, 1992.
[27] B. W. Mel, *Dendrites*, G. Stuart, N. Spruston, and M. Häusser, Eds. Oxford, England: Oxford University Press, 1999, pp. 271–289.
[28] ——, "Synaptic integration in excitable dendritic trees," *J. Neurophysiol.*, vol. 70, pp. 1086–1101, 1993.
[29] V. Peteanu, "An algebra on the optimal path in networks," *Mathematica*, vol. 9, pp. 335–342, 1967.
[30] V. Petridis and V. G. Kaburlasos, "Fuzzy lattice neural network (FLNN): A hybrid model for learning," *IEEE Trans. Neural Networks*, vol. 9, pp. 877–890, Sept. 1998.
[31] B. M. Raducanu, "Morphological techniques for face localization," Ph.D. dissertation, Univ. Basque Country, Spain, 2001.
[32] W. Rall and I. Segev, "Functional possibilities for synapses on dendrites and dendritic spines," in *Synaptic Function*, G. M. Edelman, E. E. Gall, and W. M. Cowan, Eds. New York: Wiley, 1987, pp. 605–636.
[33] G. X. Ritter and J. L. Davidson, "The image algebra and lattice theory," Univ. Florida CIS Dep., Tech. Rep. TR 87-09, 1987.
[34] ——, "Recursion and feedback in image algebra," in *Proc. SPIE 19th AIPR Workshop Image Understanding*, McLean, VA, Oct. 1990.
[35] G. X. Ritter and P. Sussner, "An introduction to morphological neural networks," in *Proc. 13th Int. Conf. Pattern Recognition*, Vienna, Austria, 1996, pp. 709–717.
[36] ——, "Associative memories based on lattice algebra," in *IEEE Int.. Conf. Syst., Man, Cybern.*, Orlando, FL, Oct. 1997, pp. 3570–3575.
[37] ——, "Morphological perceptrons," in *Proc. Intelligent Systems and Semiotics*, Gaithersburg, MD, 1997, pp. 221–226.
[38] G. X. Ritter, P. Sussner, and J. L. Diaz de Leon, "Morphological associative memories," *IEEE Trans. Neural Networks*, vol. 9, pp. 281–293, Mar. 1998.
[39] G. X. Ritter and T. Beavers, "An introduction to morphological perceptrons," in *Proc. Artificial Neural Networks Engineering*, St. Louis, MO, 1999.
[40] G. X. Ritter, J. L. Diaz de Leon, and P. Sussner, "Morphological bidirectional associative memories," *Neural Networks*, vol. 12, pp. 851–867, Mar. 1999.
[41] G. X. Ritter, G. Urcid, and L. Iancu, "Reconstruction of noisy patterns using morphological associative memories," J. Math. Imag. Vis., to be published.
[42] G. E. Robbins, M. D. Plumbey, J. C. Hughes, F. Fallside, and R. Prager, "Generation and adaptation of neural networks by evolutionary techniques (GANNET)," *Neural Comput. Applicat.*, vol. 1, pp. 23–31, 1993.
[43] A. Scott, "How smart is a neuron?," *J. Conscious. Stud.*, vol. 7, no. 5, pp. 70–75, 2000.
[44] I. Segev, "Dendritic processing," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 1998, pp. 282–289.
[45] T. J. Sejnwski and N. Qian, "Synaptic integration by electro-diffusion in dendritic spines," in *Single Neuron Computation*, T. McKenna, J. Davis, and S. F. Zornetzer, Eds. New York: Academic, 1992, pp. 117–139.
[46] R. Setiono and L. C. K. Hui, "Use of quasi-Newton method in a feedforward neural network construction algorithm," *IEEE Trans. Neural Networks*, vol. 6, pp. 273–277, Mar. 1995.
[47] R. Setiono, "A neural network construction algorithm which maximizes the likelihood function," *Connectionist Sci.*, vol. 7, pp. 147–166, 1995.
[48] ——, "Algorithmic techniques and their applications," in *Neural Network Systems Techniques and Applications*, C. T. Leondes, Ed. San Diego, CA: Academic, 1998, vol. 5, pp. 297, 301–303.
[49] G. M. Shepherd, "Canonical neurons and their computational organization," in *Single Neuron Computation*, T. McKenna, J. Davis, and S. F. Zornetzer, Eds. San Diego, CA: Academic, 1992, pp. 27–55.
[50] A. Shimbel, "Structure in communication nets," in *Proc. Symp. Information Networks*. Brooklyn, NY: Polytechnic Inst. Brooklyn, 1954, pp. 119–203.
[51] C. P. Suarez-Araujo and G. X. Ritter, "Morphological neural networks and image algebra in artificial perception systems," in *Proc. SPIE Image Algebra and Morphological Image Processing III*, vol. 1769, San Diego, CA, July 1992, pp. 128–142.
[52] C. P. Suarez-Araujo, "Novel neural network models for computing homothetic invariances: An image algebra notation," *J. Math. Imag. Vis.*, vol. 7, no. 1, pp. 69–83, 1997.
[53] P. Sussner, "Observations on morphological associative memories and the kernel method," *Neurocomputing*, vol. 31, pp. 167–183, 2000.
[54] D. S. Wei *et al.*, "Compartmentalized and binary behavior of terminal dendrites in hippocampal pyramidal neurons," *Science*, vol. 293, no. 5538, pp. 2272–2275, 2001.
[55] Y. Won and P. D. Gader, "Morphological shared weight neural network for pattern classification and automatic target detection," in *Proc. 1995 IEEE Int. Conf. Neural Networks*, Perth, Western Australia, Nov. 1995.
[56] Y. Won, P. D. Gader, and P. Coffield, "Morphological shared-weight networks with applications to automatic target recognition," *IEEE Trans. Neural Networks*, vol. 8, pp. 1195–1203, Sept. 1997.

Gerhard X. Ritter (M'83–SM'95) received the B.A. and Ph.D. degrees from the University of Wisconsin, Madison, in 1966 and 1971, respectively.

He is currently Professor of Computer Science of the Computer and Information Science and Engineering Department, the Director of the Center for Computer Vision and Visualization, and Professor of Mathematics at the University of Florida, Gainesville. He is the author of two books and more than 90 refereed publications in computer vision and mathematics. His current research interests include mathematical foundations of image processing and computer vision, and artificial neural networks.

He is the Chair of the Society of Industrial and Applied Mathematics (SIAM) Activity Group in Imaging Science and a Member of the American Association of Engineering Societies (AAES) R&D task Force. He is the Editor-in-Chief of the *Journal of Mathematical Imaging and Vision*, and a member of the Editorial Boards for both the *Journal of Electronic Imaging* and the *Journal of Pattern Analysis and Applications*. Since 1995, he has been a Fellow of SPIE and he was the recipient of the 1998 General Ronald W. Yates Award for Excellence in Technology Transfer, Air Force Research Laboratory and the 1989 International Federation for Information Processing (IFIP) Silver Core Award.

Gonzalo Urcid received the B.E. degree in 1982 and the M.Sc. degree in 1985, both from the University of the Americas, Mexico, and the Ph.D. degree in optics from the INAOE, Mexico, 1999.

He is an Associate Researcher with the Optics Department, INAOE. He holds the appointment of National Researcher from the SNI-CONACYT (National Council of Science and Technology) in Mexico, and currently is a Postdoctoral Associate in the Computer and Information Science and Engineering Department at the University of Florida. His present research interests include applied mathematics, mathematical morphology, optical information processing, and artificial neural networks.