Neurons, Dendrites, and Pattern Classification

Gerhard X. Ritter[1], Laurentiu Iancu[1], and Gonzalo Urcid[2]

[1] CISE Dept., University of Florida, Gainesville, FL 32611-6120, USA
{ritter, liancu}@cise.ufl.edu
[2] Optics Dept., INAOE, Tonantzintla, Pue. 72000, Mexico
gurcid@inaoep.mx

Abstract. Computation in a neuron of a traditional neural network is accomplished by summing the products of neural values and connection weights of all the neurons in the network connected to it. The new state of the neuron is then obtained by an activation function which sets the state to either zero or one, depending on the computed value. We provide an alternative way of computation in an artificial neuron based on lattice algebra and dendritic computation. The neurons of the proposed model bear a close resemblance to the morphology of biological neurons and mimic some of their behavior. The computational and pattern recognition capabilities of this model are explored by means of illustrative examples and detailed discussion.

1 Introduction

Various artificial neural networks (ANNs) that are currently in vogue, such as radial basis function neural networks and support vector machines, have very little in common with actual biological neural networks. A major aim of this paper is to introduce a model of an artificial neuron that bears a closer resemblance to neurons of the cerebral cortex than those found in the current literature. We will show that this model has greater computational capability and pattern discrimination power than single neurons found in current ANNs. Since our model mimics various biological processes, it will be useful to provide a brief background of the morphology of a biological neuron.

A typical neuron of the mammalian brain has two processes called, respectively, *dendrites* and *axons*. The axon is the principal fiber that forms toward its ends a multitude of branches, called the *axonal tree*. The tips of these branches, called *nerve terminals* or *synaptic knobs*, make contact with the dendritic structures of other neurons. These sites of contact are called *synaptic sites*. The synaptic sites of dendrites are the places where synapses take place. Dendrites have many branches that create large and complicated trees and the number of synapses on a *single* neuron of the cortex typically ranges between 500 and 200,000. Figure 1 provides a simplified sketch of the processes of a biological neuron. It is also well-known that there exist two types of synapses; *excitatory synapses* that play a role in exciting the postsynaptic cell to fire impulses, and

A. Sanfeliu and J. Ruiz-Shulcloper (Eds.): CIARP 2003, LNCS 2905, pp. 1–16, 2003.

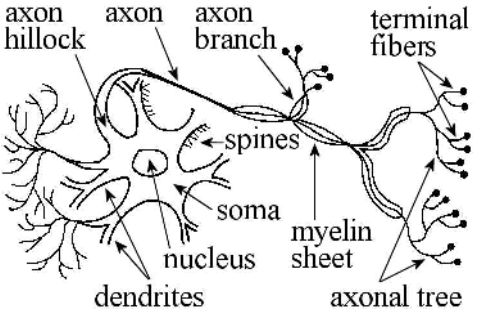

Fig. 1. Simplified sketch of the processes of a biological neuron

inhibitory synapses that try to prevent the neuron from firing impulses in response to excitatory synapses. The postsynaptic membranes of the dendrites will thus either accept or inhibit the received input from other neurons.

It is worthwhile to note that dendrites make up the largest component in both surface area *and* volume of the brain. Part of this is due to the fact that dendrites span all cortical layers in all regions of the cerebral cortex [1–3]. Thus, when attempting to model artificial brain networks, one cannot ignore dendrites, which make up more than 50% of the neuron's membrane. This is especially true in light of the fact that some researchers have proposed that dendrites, and not the neurons, are the elementary computing devices of the brain, capable of implementing such logical functions as AND, OR, and NOT [1–9].

Current ANN models, and in particular perceptrons, do not include dendritic structures. As a result, problems occur that may be easily preventable when including dendritic computing. For example, M. Gori and F. Scarselli have shown that multilayer perceptrons (MLPs) are not adequate for pattern recognition and verification [10]. Specifically, they proved that multilayer perceptrons with sigmoidal units and a number of hidden units less than or equal to the number of input units, are unable to model patterns distributed in typical clusters. The reason is that these networks draw open separation surfaces in pattern space. In this case, all patterns not members of the cluster but contained in an open area determined by the separation surfaces will be misclassified. This situation is depicted in Fig. 2. When using more hidden units than input units, the separation *may* result closed but, unfortunately, determining whether or not the perceptron draws closed separation surfaces in pattern space is NP-hard. This is quite opposite to what is commonly believed and reported in the literature. The network model described in this paper does not suffer from these problems.

Gori's and Scarselli's result was one reason for trying to use lattice algebra operations in perceptrons. Another reason is that lattice operations have proven quite successful in the area of associative memories as well as some pattern classification tasks [11–22]. Earlier attempts at morphological perceptrons did not include the notion of dendritic computing and were restricted to two-class problems. The lack of dendrites required hidden layers and computationally intensive

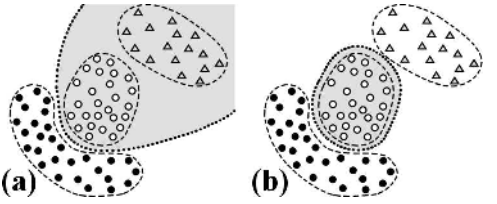

Fig. 2. In a trained MLP, the separation surface (dotted) between two clusters (black and white circles) may result open and include impostor patterns (triangles) **(a)**. A closed surface avoids this problem and is desired **(b)**

algorithms that did not provide for easy generalization to multiclass pattern separation. In contrast, the model defined in this paper uses dendritic computation, requires no hidden layers, is capable of multi-class separation to within any desired degree of accuracy, has the ability to produce closed separation surfaces between pattern clusters, and generalizes to fuzzy pattern recognition.

2 Morphological Perceptrons Based on Dendritic Computation

Let $N_1, \ldots, N_n$ denote a collection of neurons with morphology as shown in Fig. 1. Suppose these neurons provide synaptic input to another collection M_1, $\ldots$, M_m of neurons also having processes as depicted in Fig. 1. The value of a neuron N_i $(i = 1, \ldots, n)$ propagates through its axonal tree all the way to the terminal branches that make contact with the neuron M_j $(j = 1, \ldots, m)$. The weight of an axonal branch of neuron N_i terminating on the kth dendrite of M_j is denoted by w^{ℓ}_{ijk}, where the superscript $\ell \in \{0, 1\}$ distinguishes between *excitatory* $(\ell = 1)$ and *inhibitory* $(\ell = 0)$ input to the dendrite. The kth dendrite of M_j will respond to the total input received from the neurons $N_1, \ldots, N_n$ and will either accept or inhibit the received input. The computation of the kth dendrite of M_j is given by

$$\tau^j_k(\mathbf{x}) = p_{jk} \bigwedge_{i \in I(k)} \bigwedge_{\ell \in L(i)} \left(\quad 1)^{1 \quad \ell} \; x_i + w^{\ell}_{ijk} \right) \quad , \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ denotes the input value of the neurons $N_1, \ldots, N_n$ with x_i representing the value of N_i, $I(k) \subseteq \{1, \ldots, n\}$ corresponds to the set of all input neurons with terminal fibers that synapse on the kth dendrite of M_j, $L(i) \subseteq \{0, 1\}$ corresponds to the set of terminal fibers of N_i that synapse on the kth dendrite of M_j, and $p_{jk} \in \{ \quad 1, 1\}$ denotes the excitatory $(p_{jk} = 1)$ or inhibitory $(p_{jk} = \quad 1)$ response of the kth dendrite of M_j to the received input.

It follows from the formulation $L(i) \subseteq \{0, 1\}$ that the ith neuron N_i can have at most two synapses on a given dendrite k. Also, if the value $\ell = 1$, then the input $(x_i + w^1_{ijk})$ is excitatory, and inhibitory for $\ell = 0$ since in this case we have $(x_i + w^0_{ijk})$.

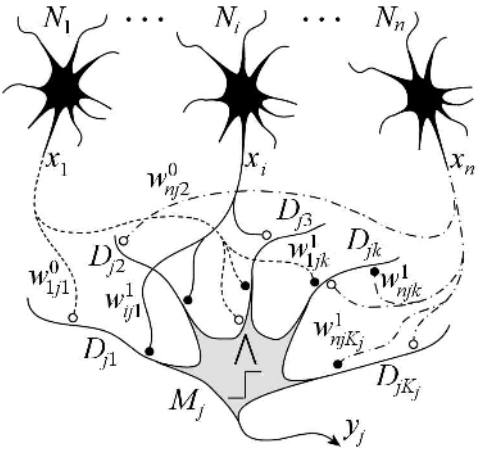

Fig. 3. Morphological perceptron with dendritic structure. Terminations of excitatory and inhibitory fibers are marked with • and ∘, respectively. Symbol D_{jk} denotes dendrite k of M_j and K_j its number of dendrites. Neuron N_i can synapse D_{jk} with excitatory or inhibitory fibers, e.g. weights w^1_{1jk} and w^0_{nj2} respectively denote excitatory and inhibitory fibers from N_1 to D_{jk} and from N_n to D_{j2}

The value $\tau^j_k(\mathbf{x})$ is passed to the cell body and the state of M_j is a function of the input received from all its dendrites. The total value received by M_j is given by

$$\tau^j(\mathbf{x}) = p_j \bigwedge_{k=1}^{K_j} \tau^j_k(\mathbf{x}) \ , \tag{2}$$

where K_j denotes the total number of dendrites of M_j and $p_j = \pm 1$ denotes the response of the cell body to the received dendritic input. Here again, $p_j = 1$ means that the input is accepted, while $p_j = \ \ 1$ means that the cell rejects the received input. The *next* state of M_j is then determined by an activation function f, namely $y_j = f\ \ \tau^j(\mathbf{x})\big)$. In this exposition we restrict our discussion to the hard-limiter

$$f\ \ \tau^j(\mathbf{x})\big) = \begin{cases} 1 \text{ if } \tau^j(\mathbf{x}) \geq 0 \\ 0 \text{ if } \tau^j(\mathbf{x}) < 0 \end{cases} \tag{3}$$

unless otherwise stated. The total computation of M_j is, therefore, given by

$$y_j(\mathbf{x}) = f\left[p_j \bigwedge_{k=1}^{K_j} \left(p_{jk} \bigwedge_{i \in I(k)} \bigwedge_{\ell \in L(i)} (\ \ 1)^{1\ \ \ell}\ x_i + w^\ell_{ijk} \right) \right] \ . \tag{4}$$

Figure 3 provides a graphical representation of this model.

A *single layer morphological perceptron* (SLMP) is a special case of this model. Here the neurons $N_i, \ldots, N_n$ would denote the input neurons and the neurons $M_1, \ldots, M_m$ the output neurons. For SLMPs we allow $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. That is, the value x_i of the ith input neuron N_i need not be binary.

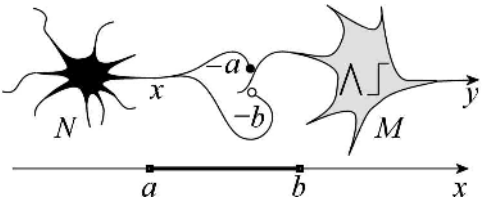

Fig. 4. The output neuron M will fire ($y = 1$) for input values from the interval $[a, b]$; if $x \in \mathbb{R} \setminus [a, b]$, then $y = 0$

3 Examples

Having defined the computational model of dendritic processes and SLMPs, it will be instructive to provide a few examples in order to illustrate the computational capabilities of the proposed model.

Example 1. The simplest case occurs when the SLMP consists of just one input neuron N, and one output neuron M with a single dendrite having an excitatory ($p_{jk} = 1$) response to the received input. Here the notation can be simplified by discarding subscripts i, j and k (as being all 1). Figure 4 illustrates such an SLMP. If the response of the cell body $p = 1$, then by (1) and (2) we have $\tau(x) = (x \quad a) \wedge \quad (x \quad b)$, and $\tau(x) \geq 0 \iff x \geq a$ and $x \leq b$. Hence, the output neuron M will fire ($y = f(\tau(x)) = 1$) when $x \in [a, b]$.

Example 2. The SLMP depicted in Fig. 5 also consists of one input neuron and one output neuron, but this time the output neuron has three dendrites, D_1, D_2 and D_3. The corresponding network parameters are given in Table 1. For algebraic consistency as well as numerical computation when using (1) and (4), *unused* terminal fibers with a hypothetical excitatory or inhibitory input will be assigned a weight of $+\infty$ or ∞ , respectively. If $a < b < c < d$, by substituting the values of the synaptic weights in (1) and (2), we obtain $\tau(x) = [(x \quad a)] \wedge \quad [(x \quad a) \wedge \quad (x \quad b)] \wedge \quad [(x \quad c) \wedge \quad (x \quad d)]$. The output neuron M will fire when $\tau(x) \geq 0 \iff x \in \{a\} \cup [b, c] \cup [d, \infty)$, as depicted on the axis at the bottom of Fig. 5.

Example 3. An SLMP with two input neurons, N_1 and N_2, and two output neurons, M_1 and M_2, can be used to solve the XOR problem, formulated as a two-class problem. Figure 6 illustrates such an SLMP. If $\mathbf{y} = (y_1, y_2)$, where y_j denotes the output signal of neuron M_j ($j = 1, 2$), then the desired network output is:

$$\mathbf{y} = \begin{cases} (1,0) \text{ if } \mathbf{x} \in C_1 \\ (0,1) \text{ if } \mathbf{x} \in C_2 \\ (0,0) \text{ if } \mathbf{x} \in \mathbb{R}^2 \setminus (C_1 \cup C_2) \ . \end{cases} \tag{5}$$

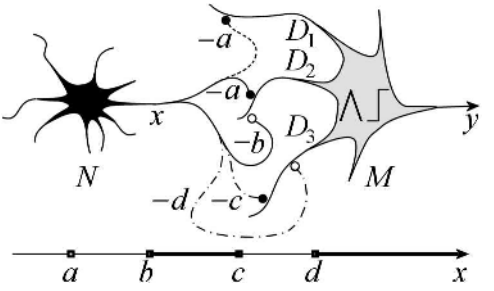

Fig. 5. The output neuron M will fire ($y = 1$) for input values from the set $X = \{a\} \cup [b, c] \cup [d, \infty)$; if $x \in \mathbb{R} \setminus X$, then $y = 0$

Table 1. Weights and Synaptic Responses, Ex. 2

D_k	w^1_{1k}	w^0_{1k}	p_k
D_1	a	∞	$+1$
D_2	a	b	1
D_3	c	d	1

As in classical perceptron theory, solving this problem requires two output neurons. However, in contrast to the classical model, no hidden layer is necessary for the morphological perceptron to solve the problem.

In this case, $\mathbf{y} = \big(f\ \ \tau^1(\mathbf{x})\big), f\ \ \tau^2(\mathbf{x})\big)\big)$, where $\tau^j(\mathbf{x}) = \bigwedge_{k=1}^{K_j} \tau^j_k(\mathbf{x})$, denotes the computation performed by M_j, and K_j denotes the number of dendrites of M_j. The values of the axonal branch weights w^ℓ_{ijk} and output responses p_{jk} are specified in Table 2.

4 Computational Capability of an SLMP

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 (4). Figure 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 non-convex region of Fig. 7. Observing differences by examples, however, does not provide answer as

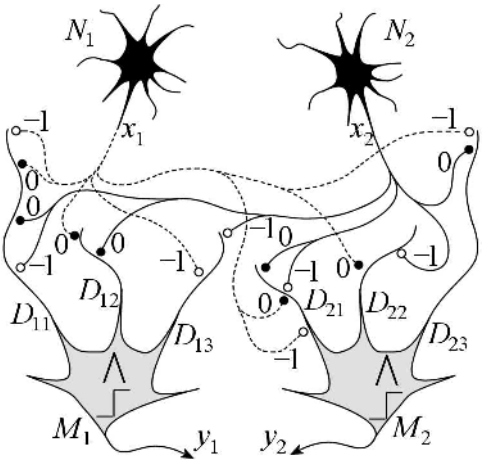

Fig. 6. SLMP that solves the two-class XOR problem for points from the domain $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$

Table 2. Two-Class XOR Network Parameters, Ex. 3

D_{jk}	w^1_{1jk}	w^0_{1jk}	w^1_{2jk}	w^0_{2jk}	p_{jk}
D_{11}	0	1	0	1	+1
D_{12}	0	∞	0	∞	1
D_{13}	$+\infty$	1	$+\infty$	1	1
D_{21}	0	1	0	1	+1
D_{22}	0	∞	$+\infty$	1	1
D_{23}	$+\infty$	1	0	∞	1

to the specific computational capabilities of an SLMP with one output neuron. Such an answer is given by the following two theorems.

Theorem 1. *If $X \subset \mathbb{R}^n$ is compact and $\varepsilon > 0$, then there exists a single layer morphological perceptron that assigns every point of X to class C_1 and every point $\mathbf{x} \in \mathbb{R}^n$ to class C_0 whenever* $\mathrm{d}(\mathbf{x}, X) > \varepsilon$.

The expression $\mathrm{d}(\mathbf{x}, X)$ in Theorem 1 refers to the distance of the point $\mathbf{x} \in \mathbb{R}^n$ to the set X. Figure 7 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 may be misclassified. As a consequence, any compact configuration, whether it is convex or non-convex, connected or not connected, contains a finite or infinite

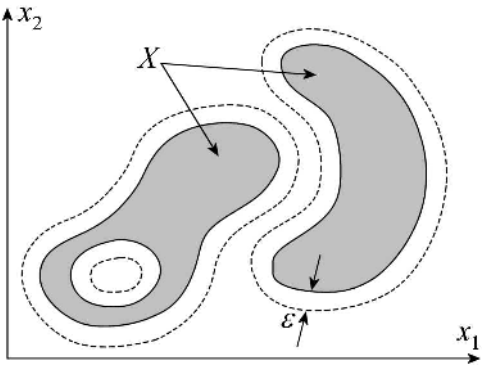

Fig. 7. The compact region X (shaded) and the banded region of thickness ε (dashed)

number of points, can be approximated to any desired degree of accuracy $\varepsilon > 0$ by an SLMP with one output neuron.

The proof of Theorem 1 requires tools from elementary point set topology and is given in [23]. Although the proof is an existence proof, part of it is constructive and provides the basic idea for our training algorithms.

Theorem 2 is a generalization of Theorem 1 to multiple sets. Suppose X_1, X_2, ..., X_m denotes a collection of disjoint compact subsets of $\mathbb{R}^n$. The goal is to classify, $\forall j = 1, \ldots, m$, every point of X_j as a point belonging to class C_j and not belonging to class C_i whenever $i \neq j$. For each $p \in \{1, \ldots, m\}$, define $Y_p = \bigcup_{j=1, j\neq p}^{m} X_j$. Since each Y_p is compact and $Y_p \cap X_p = \emptyset, \varepsilon_p = \mathrm{d}(X_p, Y_p) > 0$ $\forall p = 1, \ldots, m$. Let $\varepsilon_0 = \frac{1}{2}\min\{\varepsilon_1, \ldots, \varepsilon_p\}$.

Theorem 2. *If $\{X_1, X_2, \ldots, X_m\}$ is a collection of disjoint subsets of $\mathbb{R}^n$ and ε a positive number with $\varepsilon < \varepsilon_0$, then there exists a single layer morphological perceptron that assigns each point $\mathbf{x} \in \mathbb{R}^n$ to class C_j whenever $\mathbf{x} \in X_j$ and $j \in \{1, \ldots, m\}$, and to class $C_0 = \neg \bigcup_{j=1}^{m} C_j$ whenever $\mathrm{d}(\mathbf{x}, X_i) > \varepsilon$, $\forall i = 1, \ldots, m$. Furthermore, no point $\mathbf{x} \in \mathbb{R}^n$ is assigned to more than one class.*

Figure 8 illustrates the conclusion of Theorem 2 for the case $m = 3$. The proof of this theorem is somewhat lengthy and because of page limitation could not be included. The proof is given in [24]. Based on the proofs of these two theorems, we constructed training algorithms for SLMPs [23, 24]. During the learning phase, the output neurons grow new dendrites and input neurons expand their axonal branches to terminate on the new dendrites. The algorithms always converge and have rapid convergence rate when compared to backpropagation learning in traditional perceptrons.

These training algorithms are similar in that they all dynamically grow dendrites and axonal fibers during the learning phase, which will use the patterns of the training set in just one iteration (one *epoch*). The algorithms differ in the strategy of partitioning the pattern space, by either growing a class region by merging smaller hyper-boxes, or reducing an initial large box through elimination

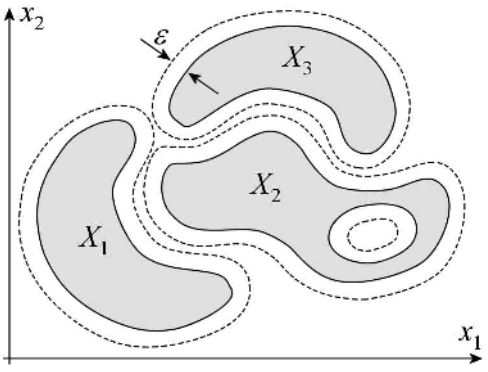

Fig. 8. The compact region X (shaded) and the banded region of thickness ε (dashed)

of foreign patterns and smaller regions that enclose them. Also, as a consequence of the aforementioned theorems on which the algorithms are based, the trained SLMPs will always correctly recognize 100% of the patterns in the training set. The next examples illustrate the performance of these training algorithms for some well-known problems.

Example 4. A nontrivial benchmark for testing the performance of a training algorithm is the well known problem of the *two intertwined spirals* [25]. For our tests, we used two *Archimedean* spirals defined by the complex expression $z_c(\theta) = x_c(\theta) + i\alpha\ \ {}^1y_c(\theta) = (\ \ 1)^1\ \ {}^c\rho\theta e^{i\theta}$, where $i = \sqrt{\ 1}$, $c \in \{0, 1\}$ denotes the spiral class label, θ is the angle in radians, $\alpha > 0$ denotes the aspect ratio between the x_c and y_c coordinates, and $\rho > 0$ is a constant that controls the spread of the spiral turns.

This problem was used to test the performance of one SLMP training algorithm that we developed, which is tailored to handle data sets where the patterns are points on a curve in 2-D space. The data set consists of 192 patterns, 96 on each spiral, 75% of which were used for training and 25% for test, selected at random from the entire set. During a typical training session, the algorithm grew 163 dendrites (161 excitatory and 2 inhibitory). Recognition of the test patterns was 100% correct. The class C_1 region learned by the SLMP is illustrated in Fig. 9. In the figure, each small rectangle represents an elementary area recognized by an individual dendrite. The solid-line rectangles correspond to excitatory dendrites; regions of inhibitory dendrites are drawn with dashed lines. The learned class C_1 region is the union of the solid-line rectangles minus the dashed-line rectangles.

Example 5. Another data set we used is the one considered in [26] to test their simulation of a radial basis function network. The data set consists of two nonlinearly separable classes of 10 patterns each. This pattern set was used as input by two other training algorithms that we developed. The former algorithm uses region merging, but assumes no a priori distribution of the patterns as did the

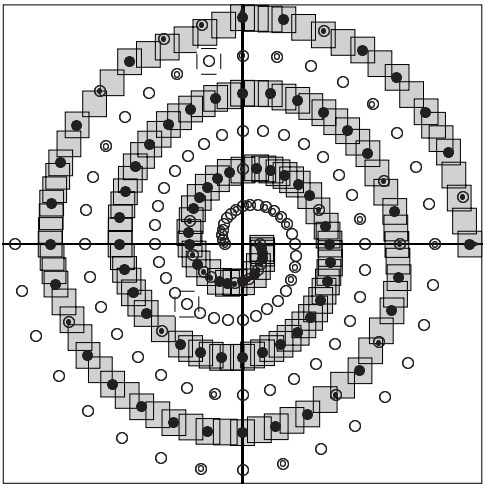

Fig. 9. The two-spirals problem. Each spiral consists of 96 patterns, 75% of which were used for training. Class C_1 patterns are marked with filled circles (training) and circled dots (test); class C_2 patterns with empty circles (training) and double circles (test). The shaded area is the learned class C_1 region; recognition is 100% correct

method mentioned in Example 4. The latter algorithm uses region elimination, i.e. it starts training by drawing an enclosing large hyper-box and proceeds by eliminating smaller regions around patterns that do not belong to the class corresponding to the enclosing hyper-box. The results of the two training algorithms are illustrated in Fig. 10 and 11, respectively. In both cases, all patterns were used for both training and test, and classification was 100% correct, as expected.

For comparison, Fig. 12 depicts the results after training a classical MLP on the same data set. The dotted lines represent the decision boundaries learned after 2000 epochs by a two-layer MLP with 13 nodes in the hidden layer using backpropagation as training algorithm. Figure 12 shows that the separation surfaces learned by the MLP are open, in contrast to the separation surfaces of SLMPs, which are guaranteed to result closed. Furthermore, convergence of the MLP is much slower than the SLMP's counterpart, even for this small data set of 20 patterns.

5 Remarks on Fuzzy Computing and Inhibitory Neurons

In our SLMP model the values of the output neurons are always crisp, i.e. having either value 1 or 0. In many application domains it is often desirable to have fuzzy valued outputs in order to describe such terms as very tall, tall, fairly tall, somewhat tall, and not tall at all. Obviously, the boundaries between these concepts cannot be exactly quantified. In particular, we would like to have output values $y_j(\mathbf{x})$ such that $0 \leq y_j(\mathbf{x}) \leq 1$, where $y_j(\mathbf{x}) = 1$ if $\mathbf{x}$ is a clear member

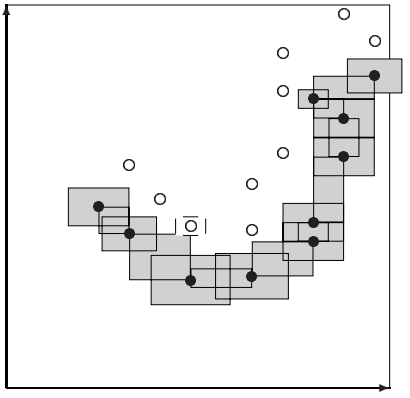

Fig. 10. The shaded are is the class C_1 region learned by the merging version of the SLMP training algorithm applied to this two nonlinearly separable classes problem. Patterns of the two classes are marked with • and ∘, respectively. The algorithms grows 20 dendrites (19 excitatory and 1 inhibitory, dashed); recognition is 100% correct

of class C_j and $y_j(\mathbf{x}) = 0$ whenever $\mathbf{x}$ has no relation to class C_j. However, we would like to say that $\mathbf{x}$ is close to full membership of class C_j the closer the value of $y_j(\mathbf{x})$ is to value 1. To illustrate how the SLMP can be extended to produce fuzzy outputs, we reconsider Example 1 of Section 3.

Suppose we would like to have every point in the interval $[a, b] \subset \mathbb{R}$ to be classified as belonging to class C_1 and every point outside the interval $[a \quad \alpha, b+\alpha]$ as having no relation to class C_1, where $\alpha > 0$ is a specified fuzzy boundary parameter. For a point $x \in [a \quad \alpha, a]$ or $x \in [b, b + \alpha]$ we would like $y(x)$ to be close to 1 when x is close to a or b, and $y(x)$ close to 0 whenever x is close to $a \quad \alpha$ or $b + \alpha$. In this case we simply convert the input $x \in \mathbb{R}$ to a new input format $\frac{x}{\alpha}$. If $w_1^0 = \quad b$ and $w_1^1 = \quad a$ denote the weights found either by inspection or the aforementioned algorithms for input x, then set $v_1^0 = \quad \frac{w_1^0}{\alpha} \quad 1$ and $v_1^1 = \quad \frac{w_1^1}{\alpha} + 1$ for the weights of the new input $\frac{x}{\alpha}$ and use the ramp activation function

$$f(z) = \begin{cases} 1 \text{ if } z \geq 1 \\ z \text{ if } 0 \leq z \leq 1 \\ 0 \text{ if } z \leq 0 \end{cases} . \tag{6}$$

Computing $\tau \quad \frac{x}{\alpha}\big)$ we obtain $\tau \quad \frac{x}{\alpha}\big) = \quad \frac{x}{\alpha} + v_1^1\big) \wedge \quad \frac{x}{\alpha} + v_1^0\big) = \big[\frac{1}{\alpha}(x \quad a) + 1\big] \wedge \big[\quad \frac{1}{\alpha}(x \quad b) + 1\big]$. Thus,

$$f\left[\tau\left(\frac{x}{\alpha}\right)\right] = \begin{cases} 1 & \text{if } x \in [a, b] \\ 0 \leq \tau \quad \frac{x}{\alpha}\big) < 1 & \text{if } x \notin [a, b] \\ 0 & \text{if } x \notin [a \quad \alpha, b + \alpha] \end{cases} . \tag{7}$$

The Equation (7) is illustrated in Fig. 13 and the network in Fig. 14. By choosing fuzzy factors α_i for each x_i, it is intuitively clear how this example generalizes to pattern vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$.

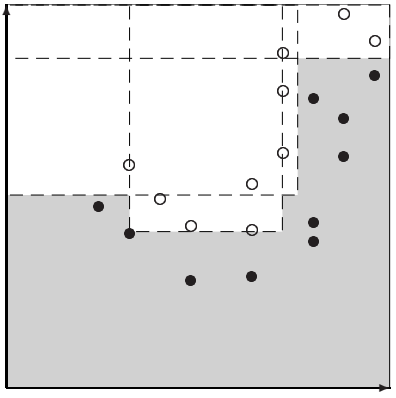

Fig. 11. The same problem as in Fig. 10, this time solved using the elimination version of the SLMP training algorithm. Only 4 dendrites of the class C_1 output neuron are sufficient to partition the pattern space similarly to the partitioning learned by a 13-hidden unit MLP, but with closed surfaces. Recognition is again 100% correct

One assumption made in our model is that a neuron N_i can provide *both* excitatory as well as inhibitory input to a neuron M_j. This assumption has no foundation in biology. In real neural networks, a neuron can send only excitatory or only inhibitory signals to other neurons. Neurons that act as inhibitors on other neurons are called *inhibitory neurons*. It is interesting to observe that all the examples given in this exposition can be expressed in terms of networks consisting entirely of excitatory and inhibitory neurons. As an illustration, let us again consider Example 1 of Section 3. In this case there are several ways of adding an inhibitory neuron. For example, we can add an inhibitory neuron N_2 so that we now have two input neurons N_1 and N_2, one sending only excitatory and the other only inhibitory inputs to the output neuron M, as shown in Fig. 15(a). In this case the axonal weights of the excitatory and inhibitory neuron are $w_1^1 = \quad a$ and $w_1^0 = \quad b$, respectively. Obviously, $\tau(x) = (x \quad a) \wedge \quad (x \quad b) \geq 0 \iff a \leq x \leq b$ and, therefore, the output of M is 1 if and only if $x \in [a, b]$. The downside of this approach is that the network topology has become a bit more complex in that we are now dealing with two input neurons. If only one input neuron is desired, then N_2 can be *partially hidden* as illustrated in Fig. 15(b). In this case, N_1 sends excitatory signals to M and N_2, and N_2 sends inhibitory signals to M. Since N_2 is not an input neuron, its states are binary, the activation function for N_2 is a hard limiter of form

$$g(z) = \begin{cases} 1 \text{ if } z > 0 \\ 0 \text{ if } z \leq 0 \end{cases} \tag{8}$$

and its axonal weight is $w_2^0 = \quad .5$. The weights of the input neuron's dendritic fibers are $w_{12}^1 = \quad a$ and $w_{21}^1 = \quad b$, which terminate on the single dendrites of N_2 and M, respectively. If $\tau(x)$ and $\tau^2(x)$ denote the total received inputs of M and N_2, then M fires if and only if $\tau(x) = (x \quad a) \wedge \left[g \quad \tau^2(x)\right) \quad 0.5] \geq 0$. Thus,

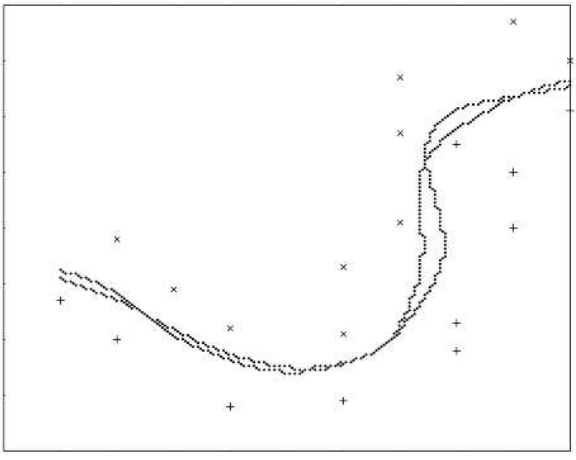

Fig. 12. Decision boundaries learned by a two-layer perceptron with 13 nodes in the hidden layer, using backpropagation. The thin space between the boundaries represents a region of uncertainty. Note that the separation surfaces are open, and compare with the regions learned by an SLMP in Fig. 10 and 11

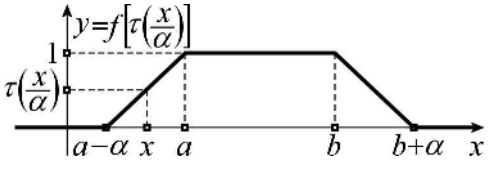

Fig. 13. Illustration of computing fuzzy output values

the output of M has value 1 if and only if $a \leq x \leq b$. Although this network also solves the problem of having only excitatory and inhibitory neurons, it is again more complex than the two-neuron model of Example 1. Even considering more complex pattern recognition problems, we have seen no *mathematical advantage* thus far in using inhibitory neurons. This does not imply that future research will not discover more powerful neural networks with dendritic structures consisting of only excitatory and inhibitory neurons. This facet of our investigations remains an active area of research.

6 Conclusions

We presented a new paradigm for neural computation that is based on lattice algebra and takes into account synaptic responses as well as computations in dendrites. The training algorithms that we developed grow new axonal fibers as well as dendritic structures during the learning phase. These facets of our model are more in agreement with current understanding of cerebral neural networks than current fashionable ANNs. The theorems that we established as

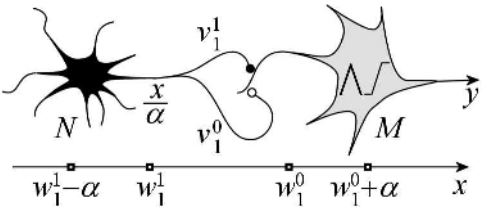

Fig. 14. The modified network of Example 1

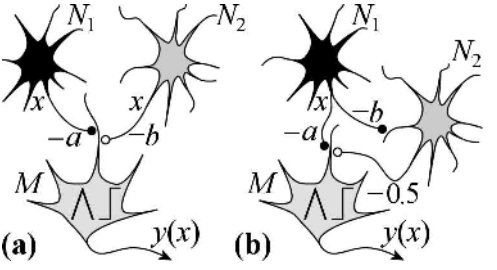

Fig. 15. Examples of two different SLMPs using only excitatory and inhibitory neurons. In (a) two input neurons are required, while in (b) a semi-hidden neuron is required

well as the examples presented in this paper make it obvious that an SLMP with just one output neuron is far more powerful as a pattern recognizer than the traditional single layer perceptron with one output neuron or a perceptron with one output neuron and one hidden layer. In fact, our training algorithms always draw a closed surface around the training set, thus preventing the problems of traditional perceptrons discussed in the Introduction.

We also indicated how our model can be generalized to include fuzzy computation. This remains an area of further research and applications. Additionally, we discussed the problem of employing only excitatory and inhibitory neurons in our model. As mentioned, this remains an active area of research and we hope that other researchers will join us in further exploration of these problems in order to bring ANNs into closer relationship with biological neural networks.

References

1. Eccles, J.C.: The Understanding of the Brain. McGraw-Hill, New York (1977)
2. Koch, C., Segev, I. (eds.): Methods in Neuronal Modeling: From Synapses to Networks. MIT Press, Boston (1989)
3. Segev, I.: Dendritic Processing. In: Arbib, M. (ed.): The Handbook of Brain Theory and Neural Networks. MIT Press, Boston (1998) 282–289
4. Arbib, M.A. (ed.): The Handbook of Brain Theory and Neural Networks. MIT Press, Boston (1998)

5. Holmes, W.R., Rall, W.: Electronic Models of Neuron Dendrites and Single Neuron Computation. In: McKenna, T., Davis, J., Zornetzer, S.F. (eds.): Single Neuron Computation. Academic Press, San Diego (1992) 7–25
6. McKenna, T., Davis, J., Zornetzer, S.F. (eds.): Single Neuron Computation. Academic Press, San Diego (1992)
7. Mel, B.W.: Synaptic Integration in Excitable Dendritic Trees. J. of Neurophysiology **70** (1993) 1086–1101
8. Rall, W., Segev, I.: Functional Possibilities for Synapses on Dendrites and Dendritic Spines. In: Edelman, G.M., Gall, E.E., Cowan, W.M. (eds.): Synaptic Function. Wiley, New York (1987) 605–636
9. Shepherd, G.M.: Canonical Neurons and their Computational Organization. In: McKenna, T., Davis, J., Zornetzer, S.F. (eds.): Single Neuron Computation. Academic Press, San Diego (1992) 27–55
10. Gori, M., Scarselli, F.: Are Multilayer Perceptrons Adequate for Pattern Recognition and Verification? IEEE Trans. on Pattern Analysis and Machine Intelligence **20**(11) (1998) 1121–1132
11. Davidson, J.L.: Simulated Annealing and Morphological Neural Networks. In: Image Algebra and Morphological Image Processing III. Proc. SPIE **1769**, San Diego, CA (July 1992) 119–127
12. Davidson, J.L., Hummer, F.: Morphology Neural Networks: An Introduction with Applications. IEEE Systems and Signal Processing **12**(2) (1993) 177–210
13. Davidson, J.L., Srivastava, R.: Fuzzy Image Algebra Neural Network for Template Identification. Second Annual Midwest Electro-Technology Conference. Ames, IA (April 1993) 68–71
14. Davidson, J.L., Talukder, A.: Template Identification Using Simulated Annealing in Morphology Neural Networks. Second Annual Midwest Electro-Technology Conference. Ames, IA (April 1993) 64–67
15. Ritter, G.X., Sussner, P.: Associative Memories Based on Lattice Algebra. IEEE Inter. Conf. Systems, Man, and Cybernetics. Orlando, FL (October 1997) 3570–3575
16. Ritter, G.X., Sussner, P., Diaz de Leon, J.L.: Morphological Associative Memories. IEEE Trans. on Neural Networks **9**(2) (March 1998) 281–293
17. Ritter, G.X., Diaz de Leon, J.L., Sussner, P.: Morphological Bidirectional Associative Memories. Neural Networks **12** (March 1999) 851–867
18. Ritter, G.X., Urcid, G., Iancu, L.: Reconstruction of Noisy Patterns Using Morphological Associative Memories. J. of Mathematical Imaging and Vision **19**(5) (2003) 95–111
19. Suarez-Araujo, C.P., Ritter, G.X.: Morphological Neural Networks and Image Algebra in Artificial Perception Systems. In: Image Algebra and Morphological Image Processing III. Proc. SPIE **1769**, San Diego, CA (July 1992) 128–142
20. Sussner, P.: Observations on Morphological Associative Memories and the Kernel Method. Neurocomputing **31**, Elsevier (2000) 167–183
21. Won, Y., Gader, P.D.: Morphological Shared Weight Neural Network for Pattern Classification and Automatic Target Detection. Proc. 1995 IEEE International Conference on Neural Networks, Perth, Western Australia (November 1995)
22. Won, Y., Gader, P.D., Coffield,P.: Morphological Shared-Weight Networks with Applications to Automatic Target Recognition. IEEE Trans. on Neural Networks **8**(5) (1997) 1195–1203
23. Ritter, G.X., Urcid, G.: Lattice Algebra Approach to Single Neuron Computation. IEEE Trans. on Neural Networks **14**(2) (March 2003) 282–295

24. Ritter, G.X., Iancu, L.: Morphological Perceptrons. Preprint submitted to IEEE Trans. on Neural Networks.
25. Lang, K.J., Witbrock, M.J.: Learning to Tell Two Spirals Apart. In: Touretzky, D., Hinton, G., Sejnowski, T. (eds.): Proc. of the 1988 Connectionist Model Summer School. Morgan Kaufmann, San Mateo, CA (1988) 52–59
26. Wasnikar, V.A., Kulkarni, A.D.: Data Mining with Radial Basis Functions. In: Dagli, C.H., et al. (eds.): Intelligent Engineering Systems Through Artificial Neural Networks. ASME Press, New York (2000)