2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Two Lattice Metrics Dendritic Computing for Pattern Recognition

Gerhard X. Ritter, Gonzalo Urcid*, and Juan-Carlos Valdiviezo-N.

Abstract— **An artificial neural network model based on dendritic computation using two lattice metrics is introduced in this paper. A description of the mathematical framework of the proposed model is provided and its corresponding learning algorithm is presented in mathematical pseudocode. Computational experiments are given to demonstrate the effectiveness and performance of the learning algorithm as well as its application to some illustrative pattern recognition problems.**

I. Introduction

Several novel approaches and techniques in artificial neural networks (ANNs), computer vision, image processing, and pattern recognition are grounded in lattice algebra [1], [2]. ANN models based on lattice operations are known as *morphological neural networks* (MNNs) or more generally as *lattice neural networks* (LNNs) and have been successfully applied to solve theoretical as well as application problems [3]–[9]. Other developments, for example, include morphological perceptrons and computational intelligence based on lattice theory [10]–[16]. In this paper, we restrict our discussion to LNNs that employ dendritic computing whose mathematical rationale was given in [11] and its biophysical motivation can be found in several works on brain theory [17]–[22].

The reason for using lattice operations when modeling dendritic computations is two-fold. First, lattice operations are extremely fast as they do not involve multiplication but only addition and the min and max operations. Second, several researchers have proposed that dendrites, and not the neurons, are the elementary computing devices of the brain, capable of implementing the logical functions AND, OR, and NOT. A simple model of a single neuron with dendritic structure using lattice operations was proposed in [11]. where it was shown that any compact region of $\mathbb{R}^n$, can be approximated to within any given degree of accuracy using a single neuron with dendritic structure. Thus, any two-class pattern recognition problem, where one class is a compact subset of $\mathbb{R}^n$ can be resolved with a single neuron. Another advantage of LNNs is the correct identification of *all* pattern vectors of the training sets after training stops. Also, during training, growth and elimination of synaptic connections and dendritic branches take place without *a priori* knowledge. Generally, these LNNs have shown superior performance when compared with several other commonly used ANNs. Nevertheless, misclassification of test data does occur and is usually due to the use of hyperboxes. The major purpose of this paper is to introduce a new model that eliminates the extreme *boxiness* and generalizes the various models derived from the original model presented in [11].

G.X. Ritter is with the CISE Department, University of Florida, FL, USA; G. Urcid (corresponding author) is with the Optics Department, INAOE, Tonantzintla, Pue., MX; J-C. Valdiviezo-N is with the Engineering Department, UPT, Tulancingo, Hgo, MX. (email's:ritter@cise.ufl.edu., gurcid@inaoep.mx, carlos.valdiviezo@upt.edu.mex)

G. Urcid and J-C. Valdiviezo-N are grateful to SNI-CONACYT for grants 22036 and 57564, respectively.

The remaining sections are arranged as follows. Section II focuses on the mathematical background that we deem necessary for a better understanding of the basic concepts of lattice group operations. Section III provides a brief review of lattice based dendritic computing and states a fundamental theorem that establishes the computational capabilities of single layer lattice perceptrons. In the same section a summary of two basic algorithmic strategies used for training is included as well as the problem associated with the *hyperbox* approach. Section IV introduces the new dendritic model based on two lattice metrics and presents simple examples that demonstrate its superiority over the original model. In the same section a learning algorithm is outlined together with illustrative pattern recognition problems. We close the paper with Section V giving the conclusions and some observations for future work.

II. Some Lattice Theory and Geometry

The computational framework for LNNs is based on lattice group operations. Here, we use the lattice groups $(\mathbb{R}, \vee, \wedge, +)$ and $(\mathbb{R}^n, \vee, \wedge, +)$, where $\mathbb{R}$ denotes the set of real numbers and $\mathbb{R}^n$ its n-fold Cartesian product so that $\mathbf{x} \in \mathbb{R}^n$ is the n-tuple $(x_1, \ldots, x_n)$ with $x_i \in \mathbb{R}$ for $i = 1, \ldots, n$. When dealing with the set $\mathbb{R}$, the binary operation $\vee$, $\wedge$, and $+$, denote the maximum, minimum, and addition of two real numbers, respectively. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{x} \vee \mathbf{y} = (x_1 \vee y_1, \ldots, x_n \vee y_n)$ and $\mathbf{x} \wedge \mathbf{y} = (x_1 \wedge y_1, \ldots, x_n \wedge y_n)$, while $+$ denotes vector addition. Occasionally it becomes convenient to use the operations of the bounded lattice ordered group, or *blog*, $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +^*)$, where $\mathbb{R}_{\pm\infty} = \mathbb{R} \cup \{-\infty, \infty\}$. Here $a \vee -\infty = -\infty \vee a = a$ and $a \wedge \infty = \infty \wedge a = a$. Similarly, $a \vee \infty = \infty \vee a = \infty$ and $a \wedge -\infty = -\infty \wedge a = -\infty$ for $a \in \mathbb{R}_{\pm\infty}$. Setting $\mathbb{R}_{\infty} = \mathbb{R} \cup \{\infty\}$ and $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$, we define $a + \infty = \infty + a = \infty +^* a = a +^* \infty = \infty$ for all $a \in \mathbb{R}_{\infty}$, and $a + (-\infty) = (-\infty) + a = (-\infty) +^* a = a +^* (-\infty) = -\infty$ for all $a \in \mathbb{R}_{-\infty}$ where $+ = +^*$ in the underlying additive group $\mathbb{R}$ of the blog (see [2] for more details).

The metrics for $\mathbb{R}^n$ that can be defined solely in terms of lattice group operations are the L_1 and L_∞ metrics. The L_1 metric, defined as $\mathrm{d}_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n |x_i - y_i|$, is known as the "city-block" or "Manhattan" distance and the L_∞ metric, defined as $\mathrm{d}_\infty(\mathbf{x}, \mathbf{y}) = \bigvee_{i=1}^n |x_i - y_i|$, is known as the *Chebyshev* or "chessboard" distance.

The set $S_p^{n-1} = \{\mathbf{x} \in \mathbb{R}^n : \mathrm{d}_p(\mathbf{0}, \mathbf{x}) = 1\}$, where $\mathbf{0}$ denotes the origin and $p \in \{1, \infty\}$, defines the $(n-1)$-

dimensional *standard unit sphere* of radius one in the metric space $(\mathbb{R}^n, \mathrm{d}_p)$. For $p = \infty$ and $n = 2$, the 1-sphere S_∞^1 ("circle"), corresponds to the boundary of the square $I^2 = \{\mathbf{x} \in \mathbb{R}^2 : -1 \leq x_i \leq 1, i = 1, 2\}$ and, for $n = 3$, the standard unit 2-sphere, S_∞^2, corresponds to the boundary of the cube $I^3 = \{\mathbf{x} \in \mathbb{R}^3 : -1 \leq x_i \leq 1, i = 1, 2, 3\}$ as shown in Fig. 1. In higher dimensions the geometry of the sphere S_∞^{n-1} remains simple since it is the boundary of the hypercube $I^n = \{\mathbf{x} \in \mathbb{R}^n : -1 \leq x_i \leq 1, i = 1, \ldots, n\}$. Equivalently, I^n denotes the compact set bounded by the $2n$ hyperplanes $x_i = -1$ and $x_i = 1$, where $i = 1, \ldots, n$.

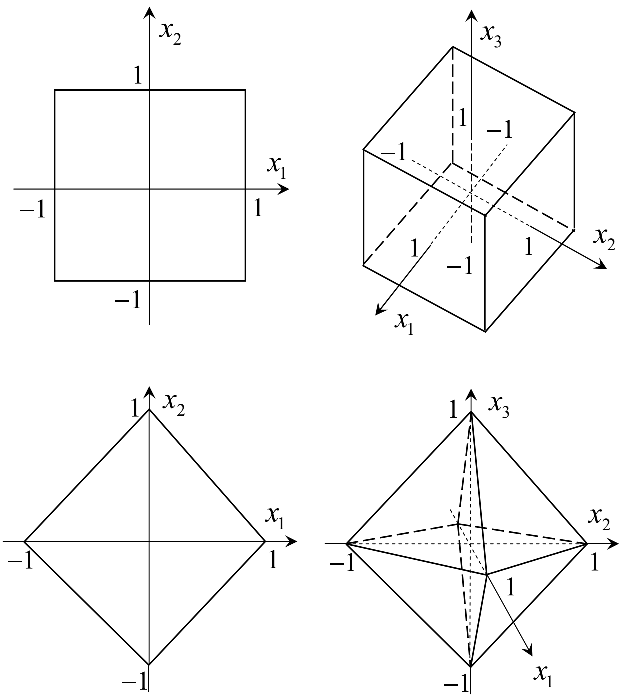

Fig. 1. Upper left: the 1-sphere $S_\infty^1 \subset \mathbb{R}^2$ (square); upper right, the 2-sphere $S_\infty^2 \subset \mathbb{R}^3$ (cube). Lower left: the 1-sphere $S_1^1 \subset \mathbb{R}^2$ (rhombus); lower right, the 2-sphere $S_1^2 \subset \mathbb{R}^3$ (octahedron).

The geometry of S_1^{n-1} and the set it bounds is more complex (see the lower part of Fig. 1 for $n = 2, 3$). The area bounded by the 1-sphere S_1^1 is the rhombus bounded by the four lines $x_1 + x_2 = -1$, $x_1 + x_2 = 1$, $x_1 - x_2 = -1$, and $x_1 - x_2 = 1$. Defining $E_1(\mathbf{x}) = x_1 + x_2$ and $E_2(\mathbf{x}) = x_1 - x_2$, it is easy to see that the set bounded by S_1^1 is given by

$$P^2 = \{\mathbf{x} \in \mathbb{R}^2 : -1 \leq E_i(\mathbf{x}) \leq 1, i = 1, 2\} \quad (1)$$

Similarly, the compact set bounded by the 2-sphere S_1^2 is the octahedron P^3 bounded by eight planes that are generated by the eight sets of affinely independent points:

$$\begin{aligned}
&V_1 = \{(1,0,0), (0,1,0), (0,0,1)\}, \\
&V_2 = \{(1,0,0), (0,1,0), (0,0,-1)\}, \\
&V_3 = \{(1,0,0), (0,-1,0), (0,0,1)\}, \\
&V_4 = \{(1,0,0), (0,-1,0), (0,0,-1)\}, \\
&V_5 = -V_1, V_6 = -V_2, V_7 = -V_3, \text{ and } V_8 = -V_4,
\end{aligned}$$

where $-V_1 = \{(-1,0,0), (0,-1,0), (0,0,-1)\}$ and for $i = 2, 3, 4$, $-V_i$ is defined in an analogous fashion. Setting $E_1(\mathbf{x}) = x_1 + x_2 + x_3$, $E_2(\mathbf{x}) = x_1 + x_2 - x_3$, $E_3(\mathbf{x}) = x_1 - x_2 + x_3$, and $E_4(\mathbf{x}) = x_1 - x_2 - x_3$, it is not difficult to establish that

$$P^3 = \{\mathbf{x} \in \mathbb{R}^3 : -1 \leq E_i(\mathbf{x}) \leq 1, i = 1, \ldots, 4\} \quad (2)$$

Equations 1 and 2 readily generalize to any dimension $n > 3$. Let $\mathbf{e}^1, \ldots, \mathbf{e}^n$ denote the standard orthonormal basis of the vector space $\mathbb{R}^n$, where $\mathbf{e}^i$ is defined by $e^i_j = 1$ if $j = i$, and $e^i_j = 0$ if $j \neq i$. Observe that the vector $\mathbf{p}^1 = (1,1) = (1,0) + (0,1) = \mathbf{e}^1 + \mathbf{e}^2$ is perpendicular to the two lines $E_1(\mathbf{x}) = \pm 1$ in Eq 1, while the vector $\mathbf{p}^2 = (1,-1) = (1,0) + (0,-1) = \mathbf{e}^1 - \mathbf{e}^2$ is perpendicular to the two lines $E_2(\mathbf{x}) = \pm 1$. Similarly, the vector $\mathbf{p}^1 = (1,1,1) = \mathbf{e}^1 + \mathbf{e}^2 + \mathbf{e}^3$ is orthogonal to the two planes $E_1(\mathbf{x}) = \pm 1$ determined by V_1 and V_5, the vector $\mathbf{p}^2 = (1,1,-1) = \mathbf{e}^1 + \mathbf{e}^2 - \mathbf{e}^3$ is orthogonal to the two planes $E_2(\mathbf{x}) = \pm 1$ determined by V_2 and V_6, the vector $\mathbf{p}^3 = (1,-1,1) = \mathbf{e}^1 - \mathbf{e}^2 + \mathbf{e}^3$ is orthogonal to the two planes $E_3(\mathbf{x}) = \pm 1$ generated by V_3 and V_7, and the vector $\mathbf{p}^4 = (1,-1,-1) = \mathbf{e}^1 - \mathbf{e}^2 - \mathbf{e}^3$ is orthogonal to the two planes $E_4(\mathbf{x}) = \pm 1$ generated by V_4 and V_8.

Next, let $\mathbb{Z}_{2^n} = \{0, 1, \ldots, 2^n - 1\}$ denote the set of integers modulo 2^n, while $\mathbb{Z}_2^n = \{\mathbf{b} : \mathbf{b} = (b_1, \ldots, b_n), b_i \in \mathbb{Z}_2, i = 1, \ldots, n\}$ denotes the set of n-dimensional binary numbers. Clearly, the two sets $\mathbb{Z}_{2^n}$ and $\mathbb{Z}_2^n$ are isomorphic. For $i, j = 1, \ldots, n$, let the function $\beta(i-1, j)$ denote the *bit value* of the j-th coordinate of the binary number $\mathbf{b}^i \in \mathbb{Z}_2^n$ representing the integer $i - 1 \in \mathbb{Z}_{2^n}$. That is,

$$\beta(i-1, j) = \mathrm{mod}\{\lfloor (i-1)/2^j \rfloor, 2\} = b^i_j \in \{0, 1\}. \quad (3)$$

The bipolar vectors $\mathbf{p}^i$, where $i = 1, \ldots, 2^{n-1}$, are defined by $p^i_j = 1$ if $b^i_j = 0$, and $p^i_j = -1$ if $b^i_j = 1$ where $j = 1, \ldots, n$. Thus, $\mathbf{p}^i = ((-1)^{\beta(i-1,1)}, \ldots, (-1)^{\beta(i-1,n)})$. If $E_i(\mathbf{x})$ is defined as the dot product $E_i(\mathbf{x}) = \mathbf{p}^i \cdot \mathbf{x}$, then setting $E_i(\mathbf{x}) = b$, where b is an arbitrary constant, results in a hyperplane with $\mathbf{p}^i$ orthogonal to this hyperplane. The n-dimensional polytope

$$P^n = \{\mathbf{x} \in \mathbb{R}^n : -1 \leq E_i(\mathbf{x}) \leq 1, i = 1, \ldots, 2^{n-1}\}, \quad (4)$$

where $E_i(\mathbf{x}) = \mathbf{p}^i \cdot \mathbf{x}$ is called the *standard star polytope*. Hypercubes and the standard star polytopes are special cases of *hyperboxes* and *star polytopes*. Given a set of constants $\{w^\ell_i : \ell \in \{0,1\}, w^1_i < w^0_i, i = 1, \ldots, n\}$, then the set

$$H^n = \{\mathbf{x} \in \mathbb{R}^n : w^1_i \leq x_i \leq w^0_i, i = 1, \ldots, n\} \quad (5)$$

is called an *n-dimensional hyperbox*. If for some non-empty subset $\{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ we have $w^1_{i_j} = w^0_{i_j}$ for $j = 1, \ldots, k$, then the set $\{\mathbf{x} \in \mathbb{R}^n : w^1_i \leq x_i \leq w^0_i, i = 1, \ldots, n\}$ is an $(n-k)$-dimensional hyperbox in $\mathbb{R}^n$. If for some $i \in \{1, \ldots, n\}$, $w^1_i = -\infty$ or $w^0_i = \infty$ (or both), then H^n is said to be *open ended at* $x_i = -\infty$ or *at* $x_i = \infty$ (or *at* $x_i = \pm\infty$), respectively. Similarly, given a set of constants $\{w^\ell_i : \ell \in \{0,1\}, w^1_i < w^0_i, i = 1, \ldots, 2^{n-1}\}$, then the set

$$P^n = \{\mathbf{x} \in \mathbb{R}^n : w^1_i \leq E_i(\mathbf{x}) \leq w^0_i, i = 1, \ldots, 2^{n-1}\} \quad (6)$$

is called a *star polytope* or L_1-*polytope*. Lower dimensional star polytopes and open endedness at $E_i(\mathbf{x})$ are defined in analogy with these concepts for hyperboxes.

III. LATTICE BASED DENDRITIC COMPUTING

Using the model of a neuron with dendritic structure as described in [11], a *single layer lattice perceptron* (SLLP) can be defined similar in structure to the classical single layer perceptron (SLP). The main differences between these two models is that the output neurons of the SLLP have dendritic structures and the neural operations are based on lattice algebraic operations. Hence, the computational capabilities of an SLLP are different from those of an SLP as well as those of multilayer perceptrons (MLPs). For example, an SLLP has no hidden layers. For the sake of completeness we summarize next the computational power of SLLPs. Suppose $X_1, X_2, \ldots, X_m$ denotes a collection of disjoint compact subsets of $\mathbb{R}^n$ and d represents one of the metrics d_1 or d_∞. The goal is to classify, for all $j = 1, \ldots, m$, every point of X_j as a point belonging to class C_j and not belonging to class C_i whenever $i \neq j$. The following result establishes the specific capabilities of SLLPs.

Theorem 3.1: Let $\mathrm{d} \in \{\mathrm{d}_1, \mathrm{d}_\infty\}$. If $\{X_1, X_2, \ldots, X_m\}$ is a collection of disjoint compact subsets of $\mathbb{R}^n$, then there exist a positive number δ such that for any positive number ε with $\varepsilon < \delta$ there exists an SLLP that assigns each point $\mathbf{x} \in \mathbb{R}^n$ to class C_j whenever $\mathbf{x} \in X_j$ and not to class C_j if $\mathrm{d}(\mathbf{x}, X_j) > \varepsilon$, where $j \in \{1, \ldots, m\}$. Furthermore, no point $\mathbf{x} \in \mathbb{R}^n$ is assigned to more than one class.

The theorem and its proof are a straight forward generalization of the two-class theorem presented in [11]. Thus, a point $\mathbf{x} \in \mathbb{R}^n$ within the ε-band surrounding X_j, but not in X_j, will be assigned to either class C_j or $C_0 = (\bigcup_{i=1}^m C_i)^c$, but not both.

Similar to the MLP, the SLLP consists of n input neurons $N_1, \ldots, N_n$, corresponding to the dimensions of the pattern vectors under consideration, and m output neurons $M_1, \ldots, M_m$, corresponding to the number of pattern classes. Each output neuron M_j has a dendritic arborization consisting of a finite number, K_j, of branches, where the k-th branch is denoted by d_{jk}, for $k \in \{1, \ldots, K_j\}$. We assume that the synapses of M_j reside on these dendritic branches. The value of a neuron N_i 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 synaptic weight associated with a synapse on the kth dendrite of M_j receiving input from a terminal axonal branch of neuron N_i is denoted by w_{ijk}^ℓ, where the superscript $\ell \in \{0, 1\}$ distinguishes between *excitatory* ($\ell = 1$) and *inhibitory* ($\ell = 0$) postsynaptic response at the synaptic site of the dendrite. Figure 2 provides a simple schematic of this setup. 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 d_{jk} of M_j is given by

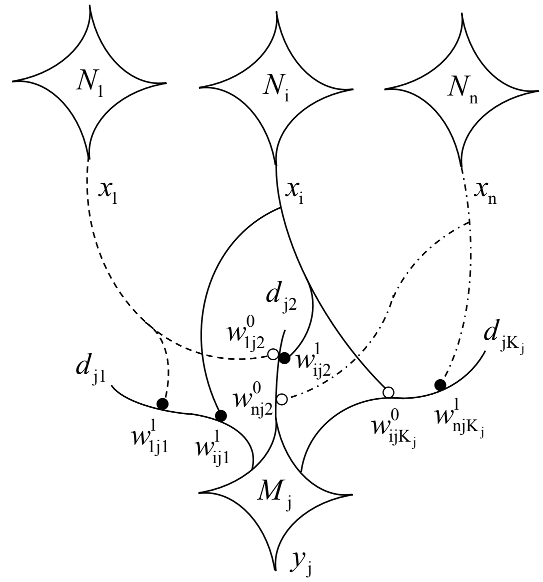

Fig. 2. An SLLP with dendritic structures. Terminal branches of axonal fibers originating from the input neurons make contact with synaptic sites on dendritic branches of M_j, denoted by d_{jk}. Synaptic sites marked by a bullet (•) correspond to synaptic weights w_{ijk}^1 and synaptic sites marked by a circle (○) correspond to weights w_{ijk}^0.

$$\tau_{kj}(\mathbf{x}) = p_{kj} \bigwedge_{i \in I_k} \bigwedge_{\ell \in L_{ik}} (-1)^{1-\ell} \left(x_i + w_{ijk}^\ell \right), \qquad (7)$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ denotes the input value for 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, i.e., the set $\{N_i : i \in I_k\}$, $L_{ik} \subseteq \{0, 1\}$ corresponds to the set of terminal fibers of N_i that synapse on the kth dendrite of M_j, and $p_{kj} \in \{-1, 1\}$ denotes the excitatory ($p_{kj} = 1$) or inhibitory ($p_{kj} = -1$) postsynaptic response of the kth dendrite of M_j to the received input.

It follows from the formulation $L_{ik} \subseteq \{0, 1\}$ 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 postsynaptic response value $(x_i + w_{ijk}^1)$ is viewed as excitatory, and inhibitory for $\ell = 0$ since in this case we have $-(x_i + w_{ijk}^0)$. In more precise mathematical terms, if N_i has *two* synapses on the kth dendrite of M_j, then the incoming information $\mathbf{x} \in \mathbb{R}^n$ will result in an excitatory post-synaptic response in the kth dendrite if and only if $(x_i + w_{ijk}^1) \wedge -(x_i + w_{ijk}^0) \geq 0$ or, equivalently, if and only if $-w_{ijk}^1 \leq x_i \leq -w_{ijk}^0$. The value $\tau_{kj}(\mathbf{x})$ is passed to the cell body and the state of M_j is a function of the input received from all its dendrites. The computed value received by M_j from its dendritic tree is given by

$$\tau_j(\mathbf{x}) = p_j \bigwedge_{k=1}^{K_j} \tau_{kj}(\mathbf{x}), \qquad (8)$$

where K_j denotes the total number of dendritic branches 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 total postsynaptic response of the cell to the received input is excitatory, while $p_j = -1$ means that the cell total postsynaptic response is inhibitory. The *next* state of M_j is determined by an activation function f, namely $y_j = f[\tau_j(\mathbf{x})]$. In this exposition we restrict our discussion to the hard-limiter

$$f[\tau_j(\mathbf{x})] = \begin{cases} 1 \Leftrightarrow \tau_j(\mathbf{x}) \geq 0 \\ 0 \Leftrightarrow \tau_j(\mathbf{x}) < 0 \end{cases} . \quad (9)$$

Variants of two approaches referred to as the *elimination* and *merge* procedures are current techniques for training SLLPs [13]–[14]. The elimination procedure takes out foreign training patterns from a region determined by a given class of training patterns, while the merging procedure builds a region of a given class of training patterns through unions of regions containing only training patterns of that class. Algorithms implementing elimination or merge techniques have many desirable properties, including fast convergence, clear geometric interpretation, 100% accurate classification of the training data, and the capability of approximating, to within any given degree of accuracy, any compact, connected or disconnected shape in Euclidean space. However, a major problem encountered by SLLP training algorithms is that many shapes cannot be modeled *exactly* or require an unreasonably large number of dendritic branches for a close approximation as explained in the next example.

Example 1: The shaded triangle in Fig. 3 is bounded by only three lines but its points cannot be classified exactly by either the elimination or merging techniques. In the left-hand illustration, the smallest rectangle enclosing all the points of the triangle is $R_1 = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_i \leq 2, i = 1, 2\}$, and after three elimination cycles, not all the white area in R_1 has been eliminated. Similarly, the right-hand side illustration shows the grey area obtained by merging 4 maximal rectangles containing only class C_1 data but not all the C_1 data.

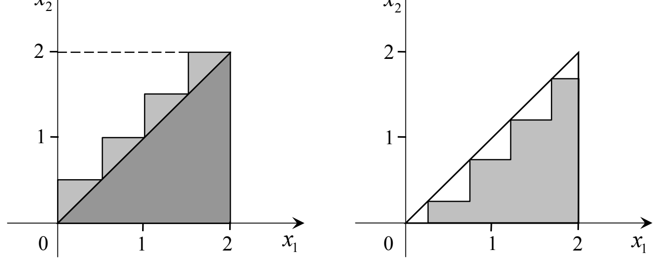

Fig. 3. If C_1 are points in the triangle and C_0 points in its complement, then the first few steps in the elimination and merging learning procedures of an SLLP are illustrated on the left and right side respectively.

The intersection $H^2 \cap P^2$ of the smallest rectangle H^2 and the smallest rectangle P^2 containing the triangular data set in Fig. 4 is exactly the triangle. Direct computation shows that $\mathbf{x} \in \mathbb{R}^2$ is a point in the triangle if and only if $0 \leq x_i \leq 2$ for $i = 1, 2$, $0 \leq E_1(\mathbf{x}) \leq 4$, and $0 \leq E_2(\mathbf{x}) \leq 2$. The corresponding lattice algebra expression satisfies the following inequality, $(2 - x_1) \wedge x_1 \wedge (2 - x_2) \wedge x_2 \wedge$ $(4 - E_1(\mathbf{x})) \wedge E_1(\mathbf{x}) \wedge (2 - E_2(\mathbf{x})) \wedge E_2(\mathbf{x}) \geq 0$, and whose left side can be written as

$$\tau(\mathbf{x}) = \bigwedge_{i=1}^{2} \bigwedge_{\ell=0}^{1} (-1)^{1-\ell}(x_i + w_i^\ell) \wedge \bigwedge_{i=1}^{2} \bigwedge_{\ell=0}^{1} (-1)^{1-\ell}(E_i(\mathbf{x}) + \omega_i^\ell), \quad (10)$$

where $w_1^0 = -2$, $w_1^1 = 0$, $w_2^0 = -2$, $w_2^1 = 0$, $\omega_1^0 = -4$, $\omega_1^1 = 0$, $\omega_2^0 = -2$, $\omega_2^1 = 0$, and $\tau(\mathbf{x}) \geq 0$ if and only if $\mathbf{x} \in H^2 \cap P^2$.

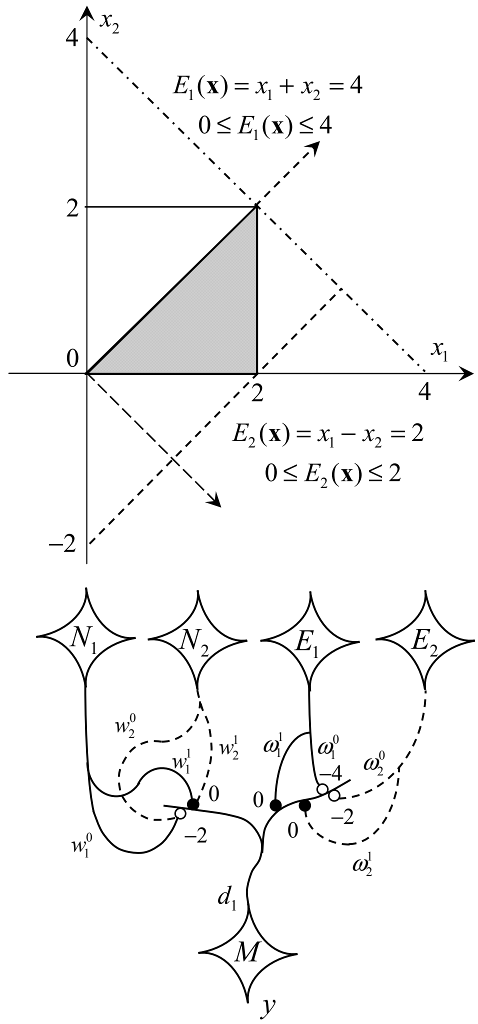

Fig. 4. The two minimal H^2 and P^2 rectangles containing the triangular region that equals $H^2 \cap P^2$ and the neural diagram representing its SLLP.

Equation 10 can be interpreted as a SLLP with four input neurons N_1, N_2, E_1, E_2, and one output neuron M having one dendrite d_1 capable of computing the value $\tau_1(\mathbf{x}) = \tau(\mathbf{x})$ for any input vector $\mathbf{x} \in \mathbb{R}^2$ (see bottom of Fig. 4).

For an input vector $\mathbf{x}$, a neuron N_i transmits the value x_i along its axonal branches to those synaptic sites on d_1 whose weight values are w_i^ℓ. In contrast, the neuron E_i produces the output $E_i(\mathbf{x}) = p^i \cdot \mathbf{x}$ and transmits it along its axon to d_1. The biological interpretation would be that E_i transmits a *spike train*, with the spikes transmitting the variables making up the expression of $E_i(\mathbf{x})$. The terminal axonal fibers of E_i impinge on the synaptic sites having weights ω_i^ℓ.

IV. The Two Lattice Metrics Model

We remark that the triangular region exhibited in Example 1 and specified by Eq. 10 serves as the motivation for the development of the $(\mathrm{d}_1, \mathrm{d}_\infty)$-*model* or *two lattice metrics model*. This model generalizes the SLLP model defined by Eqs. 7-8 and, hence, inherits all the desirable properties of the earlier model with increased classification performance. The $(\mathrm{d}_1, \mathrm{d}_\infty)$-model begins with a larger set of input neurons, namely $N_1, \ldots, N_n$, $E_1, \ldots, E_{2^{n-1}}$, and m output neurons $M_1, \ldots, M_m$, where m corresponds to the number of pattern classes under consideration. In contrast to the dendritic description given in Section 3, here each dendrite d_{jk} of M_j may have two sub-branches, denoted by d_{jk}^N and d_{jk}^E. The branch d_{jk}^N is reserved for terminal axonal fibers of N_i type neurons, while d_{jk}^E is reserved for synaptic sites of terminal axonal branches of input neuron of type E_i.

The postsynaptic responses of d_{jk}^N and d_{jk}^E are denoted by p_{kj}^N and p_{kj}^E, respectively. The total input received by the kth dendrite d_{jk} of M_j is given by

$$\tau_{kj}(\mathbf{x}) = p_{kj}^N \bigwedge_{i \in I_k^N} \bigwedge_{\ell \in L_{ki}^N} (-1)^{1-\ell}(x_i + w_{ijk}^\ell) \wedge$$
$$p_{kj}^E \bigwedge_{i \in I_k^E} \bigwedge_{\ell \in L_{ki}^E} (-1)^{1-\ell}(E_i(\mathbf{x}) + \omega_{ijk}^\ell), \qquad (11)$$

where $p_{kj}^N, p_{kj}^E \in \{-1, 1\}$, $I_k^E \subseteq \{1, \ldots, 2^{n-1}\}$ corresponds to the set of all input neurons E_i with terminal fibers that synapse on the first branch of the kth dendrite of M_j, and $L_{ik}^E \subseteq \{0, 1\}$. The sets I_k^N and L_{ik}^N have the same meaning as I_k and L_{ik} in Eq. 7. The symbol w_{ijk}^ℓ denotes the synaptic weight at synapse of N_i on d_{jk}^N and ω_{ijk}^ℓ denotes the synaptic weight at synapse of E_i on d_{jk}^E. In order to reduce the notational complexity expressed by Eq. 11, we make the following simplification. If $p_{kj}^N = p_{kj}^E$, then d_{jk} has no branching fibers. In this case we also allow the following additional simplification: If $w_{ijk}^\ell = w_{hjk}^\ell$ or $\omega_{ijk}^\ell = \omega_{hjk}^\ell$, or $w_{ijk}^\ell = \omega_{hjk}^\ell$, then the corresponding terminal axonal fibers of neurons E_i and E_h, or N_i and N_h, or, E_i and N_h (or N_i and E_h), terminate on the same synaptic site of d_{jk} as illustrated in Fig. 5. The value received by the neuron M_j is again $\tau_j(\mathbf{x})$ as defined in Eq. 8. Note that, Eq. 11 remains simple in the sense that it only involves the lattice group operations of $(\mathbb{R}, \vee, \wedge, +)$ and no multiplications. As before, the value $\tau_{kj}(\mathbf{x})$ is passed to the cell body of M_j. The total information computed by the dendrites is combined by M_j using the formulation $\tau_j(\mathbf{x}) = p_j \bigwedge_{k=1}^{K_j} \tau_{kj}(\mathbf{x})$ where $p_j \in \{-1, 1\}$ denotes the postsynaptic response of the neuron.

Example 2: The 3-dimensional XOR problem provides an excellent visual and algebraic comparison with the hyperbox elimination algorithm presented in [11]. In the XOR problem we assume all patterns under consideration are boolean, i.e., patterns are elements of $\mathbb{Z}_2^3$ where $\mathbb{Z}_2 = \{0, 1\}$. For a point $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{Z}_2^3$ define its index ξ by $\xi = 4x_1 + 2x_2 + x_3 + 1$, so that $\mathbf{x}^1 = (0, 0, 0)$, $\mathbf{x}^2 = (0, 0, 1), \ldots, \mathbf{x}^8 = (1, 1, 1)$. The class pattern of $\mathbf{x}^\xi = (x_1^\xi, x_2^\xi, x_3^\xi)$, for $\xi = 1, \ldots, 8$, is defined in terms of the exclusive or operation $\oplus$ of its coordinates, $c_\xi = x_1^\xi \oplus x_2^\xi \oplus x_3^\xi$. Specifically, class C_1 is given by $C_1 = \{\mathbf{x}^\xi \in \mathbb{Z}_2^3 : c_\xi = 1\} = \{\mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^5, \mathbf{x}^8\}$ while $C_0 = \{\mathbf{x}^\xi \in \mathbb{Z}_2^3 : c_\xi = 0\} = \{\mathbf{x}^1, \mathbf{x}^4, \mathbf{x}^6, \mathbf{x}^7\}$. In this case, only one output neuron with a dendrite is required ($j, k = 1$), hence the corresponding lattice algebra expression based on Eq. 11 is simplified to

$$\tau(\mathbf{x}) = \bigwedge_{i \in I^E} \bigwedge_{\ell \in L_i^E} (-1)^{1-\ell}(E_i(\mathbf{x}) + w_i^\ell) = (E_1(\mathbf{x}) - 1) \wedge$$
$$(1 - E_2(\mathbf{x})) \wedge (1 - E_3(\mathbf{x})) \wedge (E_4(\mathbf{x}) + 1), \qquad (12)$$

where $L_1^E = L_4^E = \{1\}$, $L_2^E = L_3^E = \{0\}$, $\omega_1^1 = \omega_2^0 = \omega_3^0 = -1$, and $\omega_4^1 = 1$. Using the algorithm given in [11] results in a net consisting of three input neurons and one output neuron with 5 dendrites and 18 synaptic sites. In contrast, Eq. 12 yields a network with four input neurons and one output neuron with a single dendrite and 3 synaptic sites as shown in Fig. 5. The 3-D XOR problem is equivalent to the n-parity problem with $n = 3$ and it has been pointed out that a dynamic node creation algorithm for feed-forward networks needs 2 or 3 neurons in a single hidden layer in order to correctly classify all inputs from $\mathbb{Z}_2^3$ [23], [24].

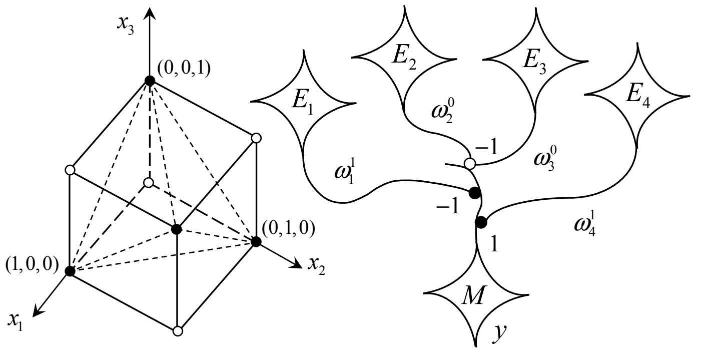

Fig. 5. In the 3-D XOR problem the class C_1 and class C_0 points are represented by the solid and by the hollow points, respectively. Here the smallest box H^3 containing C_1 is the cube determined by the points of $C_0 \cup C_1$ while the smallest polyhedron P^3 containing C_1 is determined by the four triangles whose vertices are all C_1 points. These vertices are the only 3-D binary points that will provide non-negative output when used as input to the net on the right side.

Training in the two metrics model means the creation of neural connections in terms of axonal fibers, dendritic structures, and synapses that are grown and modified through feedback loops triggered whenever a training pattern is misclassified. The learning process is tailored to build a network that provides 100% correct classification of training patterns. Here we generalize the *elimination* algorithm presented in [14] by incorporating both, the L_∞ and L_1 metrics. The first part of the training algorithm is modeled after the two class problem given in [11]. In the following we let $T = \{\mathbf{x}^\xi \in \mathbb{R}^n : \xi = 1,\dots,t\}$ denote the training set for an m class problem $C_1,\dots,C_m$, $P = \{1,\dots,t\}$, $T_j = \{\mathbf{x}^\xi \in T : \mathbf{x}^\xi \in C_j\}$ for $j = 1,\dots,m$, and $c_j^\xi \in \{0,1\}$, where $c_j^\xi = 1$ if $\mathbf{x}^\xi \in T_j$ and $c_j^\xi = 0$ if $\mathbf{x}^\xi \notin T_j$. Also, $T_j^c = T \setminus T_j$. The sets $I_k^N, L_{ki}^N, I_k^E, L_{ki}^E$ appearing in Eq. 11 are generated during training.

The following values are pre-established: for $\ell = 0,1$ and $j = 1,\dots,m$ compute $\varepsilon_j^N = \bigwedge_{\mathbf{x}\in T_j} \mathrm{d}_\infty(\mathbf{x}, T_j^c)$, where $\mathrm{d}_\infty(\mathbf{x}, T_j^c) = \bigwedge_{\mathbf{y}\in T_j^c} \mathrm{d}_\infty(\mathbf{x},\mathbf{y})$, and $\delta_j^{\ell N} = (-1)^{1-\ell}\alpha_j^N \varepsilon_j^N$. Similarly, $\varepsilon_j^E = \bigwedge_{\mathbf{x}\in T_j} \mathrm{d}_1(\mathbf{x}, T_j^c)$, where $\mathrm{d}_1(\mathbf{x}, T_j^c) = \bigwedge_{\mathbf{y}\in T_j^c} \mathrm{d}_1(\mathbf{x},\mathbf{y})$, and $\delta_j^{\ell E} = (-1)^{1-\ell}\alpha_j^E \varepsilon_j^E$. The α_j^N, α_j^E parameters are user defined and restricted to $[0, 0.5)$. For $\alpha_j^N = 0$ or $\alpha_j^E = 0$ some elements of T_j will lie on the boundary of the region recognized by M_j. Restricting α_j^N, α_j^E to $(0, 0.5)$ aids in reducing possible overlap between classes and will ensure that no training pattern will lie on the boundary of the region recognized by M_j. The learning algorithm is presented below in numbered steps prefixed by S and brief comments are provided within brackets.

ALGORITHM: $L_\infty L_1$-SLLP Training by Elimination

S0 $j = 0$ [Initialize class counter]

S1 $j = j + 1$
  if $j > m + 1$ then **stop**
  else $k = 1$; $I_k^N = \{1,\dots,n\}$; $I_k^E = \{1,\dots,2^{n-1}\}$

[If training for all output neurons is not complete, initialize auxiliary sets for the creation of the first dendrite of M_j]

S2 **for** $i \in I_k^N$
  $w_{ijk}^0 = [-\bigvee_{c_j^\xi=1} x_i^\xi)] + \delta_j^{0,N}$
  $w_{ijk}^1 = [-\bigwedge_{c_j^\xi=1} x_i^\xi] + \delta_j^{1,N}$; $L_{ki}^N = \{0,1\}$
  for $i \in I_k^E$
  $\omega_{ijk}^0 = [-\bigvee_{c_j^\xi=1} E_i(\mathbf{x}^\xi)] + \delta_j^{0,E}$
  $\omega_{ijk}^1 = [-\bigwedge_{c_j^\xi=1} E_i(\mathbf{x}^\xi)] + \delta_j^{1,E}$; $L_{ki}^E = \{0,1\}$

[Compute weights for synaptic sites of first dendrite that determine $H^n \cap P^n$ enclosing training class T_j; k is the dendrite counter]

S3 $p_{kj}^N = p_{kj}^E = (-1)^{\mathrm{sgn}(k-1)}$
  for $\xi = 1$ to t

$$\tau_{kj}^N(\mathbf{x}^\xi) = p_{kj}^N \bigwedge_{i\in I_k^N} \bigwedge_{\ell\in L_{ki}^N} (-1)^{1-\ell}(x_i^\xi + w_{ijk}^\ell)$$

$$\tau_{kj}^E(\mathbf{x}^\xi) = p_{kj}^E \bigwedge_{i\in I_k^E} \bigwedge_{\ell\in L_{ki}^E} (-1)^{1-\ell}(E_i(\mathbf{x}^\xi) + \omega_{ijk}^\ell)$$

$$\tau_{jk}(\mathbf{x}^\xi) = \tau_{kj}^N(\mathbf{x}^\xi) \wedge \tau_{kj}^E(\mathbf{x}^\xi)$$

$$\tau_j(\mathbf{x}^\xi) = \bigwedge_{h=1}^{k} \tau_h^j(\mathbf{x}^\xi)$$

[Using all training patterns, compute partial responses for each lattice metric and total response of current dendrite d_{jk} of output neuron M_j; sgn is the signum function]

S4 **for** $\xi = 1$ to t
  $R_k = \{\mathbf{x}^\xi \in T : f(\tau^j(\mathbf{x}^\xi)) = 1\}$
  if $R_k \setminus T_j = \varnothing$ [or, if $f(\tau^j(\mathbf{x}^\xi)) = c_j^\xi,\ \forall \xi \in P$]
    $K_j = k$ and **return** to S1
  else randomly choose $\mathbf{x}^\gamma \in R_k \setminus T_j$

[If training for M_j is successful (using the hard limiter f), then the final number of dendrites grown by neuron M_j is K_j and the training of neuron M_{j+1} can commence, else randomly select a misclassified pattern $\mathbf{x}^\gamma$]

S5 $k = k + 1$
  $I_k^N = I_k^{0,N} = I_k^{1,E} = \{1,\dots,n\}$
  $I_k^E = I_k^{0,E} = I_k^{1,E} = \{1,\dots,2^{n-1}\}$
  for $i \in I_k^N$, $L_{ki}^N = \{0,1\}$
  for $i \in I_k^E$, $L_{ki}^E = \{0,1\}$

[Add a new dendrite k to M_j; initialize its indexing sets]

S6 **for** $i \in I_k^N$, $p, q = 0$; $s_p, t_q = 0$
  for $\mathbf{x}^\xi \in T_j$
  if $x_i^\xi > x_i^\gamma$, then $p = p + 1, s_p = \xi$, **else** continue
  if $x_i^\xi < x_i^\gamma$, then $q = q + 1, t_q = \xi$, **else** continue
  if $p > 0$
    $d = \bigwedge_{\lambda=1}^p \mathrm{d}_\infty(\mathbf{x}^\gamma, \mathbf{x}^{s_\lambda})$; $w_{ijk}^0 = -(x_i^\gamma + \frac{1}{2}d)$
  else $I_k^{0,N} = I_k^{0,N} \setminus \{i\}$; $L_{ki}^N = L_{ki}^N \setminus \{0\}$
  if $q > 0$
    $d = \bigwedge_{\lambda=1}^q \mathrm{d}_\infty(\mathbf{x}^\gamma, \mathbf{x}^{s_\lambda})$; $w_{ijk}^1 = -(x_i^\gamma - \frac{1}{2}d)$
  else $I_k^{1,N} = I_k^{1,N} \setminus \{i\}$; $L_{ki}^N = L_{ki}^N \setminus \{1\}$
  $I_k^N = I_k^{0,N} \cup I_k^{1,N}$

[Cycle through input neurons of type N and assign L_∞ metric weights to new synaptic sites that may classify $\mathbf{x}^\gamma$ with the current dendrite d_{jk} of output neuron M_j]

S7 **for** $i \in I_k^E$, $p, q = 0$; $s_p, t_q = 0$
  for $\mathbf{x}^\xi \in T_j$
  if $E_i(\mathbf{x}^\xi) > E_i(\mathbf{x}^\gamma)$, then $p = p + 1, s_p = \xi$
    else continue
  if $E_i(\mathbf{x}^\xi) < E_i(\mathbf{x}^\gamma)$, then $q = q + 1, t_q = \xi$
    else continue
  if $p > 0$
    $d = \bigwedge_{\lambda=1}^p \mathrm{d}_1(\mathbf{x}^\gamma, \mathbf{x}^{s_\lambda})$; $\omega_{ijk}^0 = -(E_i(\mathbf{x}^\gamma) + \frac{1}{2}d)$
  else $I_k^{0,E} = I_k^{0,E} \setminus \{i\}$; $L_{ki}^E = L_{ki}^E \setminus \{0\}$
  if $q > 0$
    $d = \bigwedge_{\lambda=1}^q \mathrm{d}_1(\mathbf{x}^\gamma, \mathbf{x}^{s_\lambda})$; $\omega_{ijk}^1 = -(E_i(\mathbf{x}^\gamma) - \frac{1}{2}d)$
  else $I_k^{1,E} = I_k^{1,E} \setminus \{i\}$; $L_{ki}^E = L_{ki}^E \setminus \{1\}$
  $I_k^E = I_k^{0,E} \cup I_k^{1,E}$; **loop** back to S3

[Cycle through input neurons of type E and assign L_1 metric weights to new synaptic sites that may classify $\mathbf{x}^\gamma$ with the current dendrite d_{jk} of output neuron M_j]

It is important to realize that the training algorithm can be modified to include a priory, probabilistic, or statistical knowledge derived from the data or training set under consideration. Thus, e.g., the parameters δ_j^ℓ associated to neurons of type N or E, are independent of the coordinate index i but could be generalized to δ_{ij}^ℓ by including knowledge obtained from the spatial distribution of the ith coordinates x_i^ξ of the training data. Similarly, when dealing strictly with n-dimensional Boolean patterns, then choosing $\alpha_j > 0$ (for the N's or E's) is not very helpful since all patterns under consideration are elements of the set $B^n = \{\mathbf{x} \in \mathbb{R}^n : x_i \in \mathbb{Z}_2, i = 1, \ldots, n\}$. In this case setting $\alpha_j = 0$ is, generally, the best choice. We remind that if the network output class matches the known class of a given test pattern a hit is obtained (correct classification), otherwise a misclassification error occurs. Hence, the proposed $L_\infty L_1$-SLLP recognition capability is measured by computing the fraction of hits relative to each input set used for testing.

Thus, given a data set, $X = \bigcup_{j=1}^m X_j$, a family of training subsets, denoted by T_{jp}, were generated by randomly selecting predefined percentages $p\%$ of the total number k_j of samples in X_j for $j = 1, \ldots, m$. Specifically, p percentages where considered in the range $\{10\%, 20\%, \ldots, 90\%\}$ and the corresponding test subsets are given by $T_{jp}^c = X_j \backslash T_{jp}$. Also, a finite number of runs were realized in order to compute the *overall class* average fraction of hits for each selected percentage of all samples. Thus, if $|X| = k = \sum_{j=1}^m k_j$, $|T_p| = \sum_{j=1}^m T_{jp}$, $|T_p^c| = \sum_{j=1}^m T_{jp}^c$, represent the set cardinality of data, training, and test sets, respectively, τ is the number of runs, ρ_r is the number of correctly classified test patterns, and μ_r denotes the number of misclassified test patterns in each run, then the *average fraction of hits* is given by,

$$\bar{f}_p = 1 - \frac{\bar{\mu}}{k} \; ; \; \bar{\mu} = \frac{1}{\tau}\sum_{r=1}^{\tau} \mu_r \; ; \; \bar{\rho} = \frac{1}{\tau}\sum_{r=1}^{\tau} \rho_r \, , \qquad (13)$$

and $k = |T_p| + |T_p^c|$. In (13), we set $\tau = 20$ and use the same value for any p. Though T_p and T_p^c have the same number of elements for each run with the same value of p, the sample points belonging to each subset are different since they are randomly generated.

The results of our computer experiments for $L_\infty L_1$-SLLP learning and classification of patterns in the following example data sets are displayed in table format and $\alpha_j^N = \alpha_j^E = 0.49$ for all j. The 1st column gives the percentage p of sample points used to build the training and test subsets, the 2nd column provides the average number of correctly classified data patterns using the combined L_∞ and L_1 lattice metrics, the 3rd column gives the average number of misclassified inputs, the 4th and following columns shows the average number of dendrites per class, $\bar{K}_j$ for $j = 1, \ldots, m$, as generated by the elimination algorithm, and the last column gives the average fraction of hits.

Example 3. Here, set X consists of 294 samples equidistributed in two classes C_j with $j = 1, 2$, and forming a 'Twins' shape in the plane whose features are the x and y coordinates. The corresponding two-dimensional point set is shown in Fig. 6 and Table I gives the numerical results.

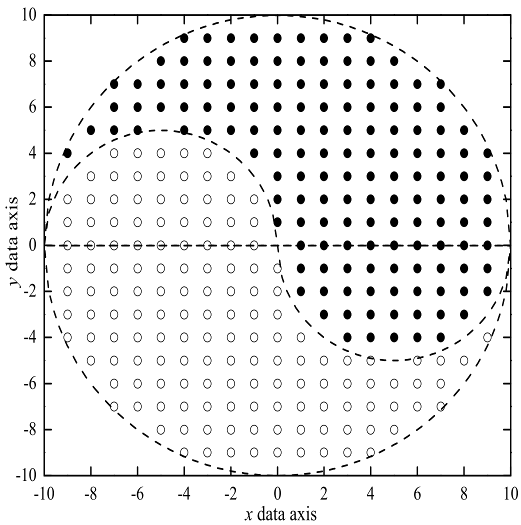

Fig. 6. A discrete subset X of $\mathbb{R}^2$. Points belonging to class C_1 are marked by a circle (○) and points of class C_2 are marked with a disk (●).

TABLE I
L_∞-L_1 SLLP CLASSIFICATION PERFORMANCE FOR THE 'TWINS' SET; 294 SAMPLES (k), 2 FEATURES (n), 2 CLASSES (m)

p	$\bar{\rho}$	$\bar{\mu}$	$\bar{K}_1$	$\bar{K}_2$	$\bar{f}_p$
10%	241	53	3	2	0.820
20%	263	31	9	6	0.895
⋆22%	294	0	21	21	1.000
30%	267	27	17	7	0.908
40%	278	16	18	11	0.945
50%	279	15	20	18	0.949
60%	283	11	14	25	0.963
70%	286	8	15	34	0.973
80%	288	6	34	32	0.980
90%	291	3	43	41	0.990

Note that the entry in Table I marked with a star (⋆) corresponds to a training subset composed of 64 samples selected by *border points* delimiting each class, and our algorithm provides correct classification for any input test pattern. In this instance, the $L_\infty L_1$-SLLP shows the advantage of using problem-related knowledge before learning. Also, observe that the number of dendrites required is much less than those required for a training subset with 80% or 90% of the patterns (cf. last two rows of Table I).

Example 4. Here we use the 'Iris' data set [25]-[26] with 150 samples equally distributed in 3 classes corresponding, respectively, to the subspecies of Iris setosa (C_1), Iris versicolor (C_2), and Iris virginica (C_3). Each sample is described by four flower features: sepal length, sepal width, petal length, and petal width. Table II displays the classification results.

TABLE II

L_∞-L_1 SLLP CLASSIFICATION PERFORMANCE FOR THE 'IRIS' SET; 150 SAMPLES (k), 4 FEATURES (n), 3 CLASSES (m)

p	$\bar{\rho}$	$\bar{\mu}$	$\bar{K}_1$	$\bar{K}_2$	$\bar{K}_2$	$\bar{f}_p$
10%	92	58	1	1	1	0.613
20%	126	24	1	2	1	0.840
30%	134	16	1	1	2	0.893
$\star$38%	150	0	1	7	10	1.000
40%	134	16	1	3	2	0.893
50%	140	10	1	3	3	0.933
60%	143	7	1	5	6	0.953
70%	145	5	1	6	5	0.960
80%	147	3	1	6	7	0.980
90%	148	2	1	6	12	0.987

Note that for the data sets 'Twins' and 'Iris', a high average fraction of hits such as $\bar{f}_p > 0.95$ is obtained for percentages p as low as 60%. Furthermore, the entry in Table II marked with a star ($\star$) corresponds to a training subset composed of 57 samples selected from an ascending *lexicographic ordering* within each class, and our algorithm provides correct classification for any input test pattern. Notice, in particular, that the number of dendrites required is practically the same as if 90% of the patterns were used for training (see last row in Table II). Hence, in this case the $L_\infty L_1$-SLLP used as an individual classifier delivers better performance, e.g., against Linear or Quadratic Bayesian classifiers [27] for which, $\bar{f}_{50} = 0.953$ and $\bar{f}_{50} = 0.973$ (with $\tau = 2$), respectively, or in comparison with an Edge-effect Fuzzy Support Vector Machine [28] whose $\bar{f}_{60} = 0.978$ (here, τ is not specified explicitly).

V. CONCLUSIONS

This paper introduces an extended training algorithm for neural networks endowed with dendrites based on the lattice metrics L_∞ and L_1. The mathematical background, the $L_\infty L_1$-SLLP model description, and the mathematical pseudo-code of the proposed learning by elimination algorithm is given together with illustrative non-linear examples that demonstrates its performance capabilities under random sampling or problem-related knowledge. Future work contemplates algorithm optimization by pruning redundant neuron to dendrite connections, additional computer experiments using higher-dimensional problems in pattern classification, and performance comparisons with other current techniques in the field of machine learning.

REFERENCES

[1] G. Birkhoff, *Lattice Theory*, American Mathematical Society Colloquium Publications, Vol. 25, Providence, RI, 1940.

[2] G.X. Ritter and P. Gader, "Fixed Points of Lattice Transforms and Lattice Associative Memories," in P. Hawkes (Ed.), *Adv. in Imaging & Electron Physics*, Elsevier, San Diego, CA, Vol. 144, 165–242, 2006.

[3] V. Petridis and V.G. Kaburlasos, "Fuzzy lattice neural network (FLNN): a hybrid model for learning," *IEEE Transactions on Neural Networks*, 9, 877–890, 1998.

[4] M.A. Khabou, P.D. Gader, and J.M. Keller, "LADAR target detection using morphological shared-weight neural networks," *Machine Vision and Applications*, 11(6) 300–305, 2000.

[5] L.F.C. Pessoa and P. Maragos, "Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition," *Pattern Recognition*, 33(6), 945–960, 2000.

[6] P. Maragos, "Lattice image processing: A unification of morphological and fuzzy algebraic systems," *Journal of Mathematical Imaging and Vision*, 22(2-3), 333–353, 2005.

[7] G.X. Ritter, G. Urcid, and L. Iancu, "Reconstruction of Noisy Patterns Using Morphological Associative Memories," *Journal of Mathematical Imaging and Vision*, 19(2), 95–111, 2003.

[8] V.G. Kaburlasos, "Improved Fuzzy Lattice Neurocomputing (FLN) for Semantic Neural Computing," *IEEE Proceedings of IJCNN*, Portland, OR, 1850–1855, 2003.

[9] M. Grana, I. Villaverde, J.O. Maldonado, and C. Hernandez, "Two lattice approaches for unsupervised segmentation of hyperspectral images," *Neurocomputing*, 72(10-12), 2111–2120, 2009.

[10] G.X. Ritter, L. Iancu, and G. Urcid, "Morphological perceptrons with dendritic structure," *IEEE Proceedings of FUZZ'03*, St. Louis, MO, 1296–1301, 2003.

[11] G.X. Ritter and G. Urcid, "Lattice algebra approach to single-neuron computation," *IEEE Transactions on Neural Networks*, 14(2), 282–295, 2003.

[12] G. Urcid, G.X. Ritter, and L. Iancu, "Single layer morphological perceptron solution to N-bit parity problem," in LNCS, Vol. 3287, Springer, Berlin-Heidelberg, 163–170, 2004.

[13] L. Iancu, *Lattice Algebra Approach to Neural Computation*, Ph.D. Thesis, University of Florida, Gainesville, FL, 2005.

[14] G.X. Ritter and G. Urcid, "Learning in Lattice Neural Networks that Employ Dendritic Computing," in *Computational Intelligence based on Lattice Theory*, Vol. 67, V.G. Kaburlasos, G.X. Ritter (Eds.), Springer, Berlin, 25–44, 2007.

[15] V.G. Kaburlasos and G.X. Ritter (Eds.), *Computational Intelligence Based on Lattice Theory*, Vol. 67. Springer, Berlin, 2007.

[16] H. Sossa, and E. Guevara, "Efficient training for dendrite morphological neural networks," *Neurocomputing*, 131, 132–142, 2013.

[17] C. Koch and I. Segev (Eds.), *Methods in Neuronal Modeling: From Synapses to Networks*, MIT Press, Boston, 1989.

[18] W.R. Holmes and W. Rall, "Electronic Models of Neuron Dendrites and Single Neuron Computation," in *Single Neuron Computation*, T. McKenna, J. Davis, and F. Zornetzer (Eds.), Academic Press, New York, 7–25, 1992.

[19] G.M. Shepherd, "Canonical Neurons and their Computational Organization," in *Single Neuron Computation*, T. McKenna, J. Davis, and F. Zornetzer (Eds.), Academic Press, New York, 27–55, 1992.

[20] B.W. Mel, "Synaptic Integration in Excitable Dendritic Trees," *Journal of Neurophysiology*, 70, 1086–1101, 1993.

[21] I. Segev, "Dendritic Processing," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib (Ed.), MIT Press, Boston, 282–289, 1998.

[22] B.W. Mel,"Why have Dendrites? A Computational Perspective," in *Dendrites*, G. Stuart, N. Spruston, and M.D. Hausser (Eds.), Oxford University Press, 271–280, 1999.

[23] R. Setiono, "A Neural Network Construction Algorithm which maximizes the Likelihood Function," *Connectionist Science*, 7, 147–166, 1995.

[24] R. Setiono, "Algorithm Techniques and their Applications," in C.T. Leondes (Ed.), *Neural Network Systems Techniques and Applications*, Vol. 5, Academic Press, San Diego, CA., 297–303, 1998.

[25] J.C. Bezdek, J. Keller, R. Krisnapuram, N.R. Pal, "Cluster Analysis for Object Data," in J.C. Bezdek, et al. (Eds.), *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer Academic, Dordrecht, The Netherlands, 87–121, 1999.

[26] A. Frank and A. Asuncion, *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml]. University of California, School of Information & Computer Science, Irvine, CA., 2010.

[27] K. Woods, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), 405–410, 1997.

[28] C-F. Li, L. Xu, and S-T. Wang, "A comparative study on improved fuzzy support vector machines and Levenberg-Marquardt based BP network," *Intelligent Computing*, Springer LNCS, Vol. 4113, 73–82, Berlin, 2006.