Two Lattice Metrics Dendritic Computing for Pattern Recognition

Gerhard X. Ritter, Gonzalo Urcid\textsuperscript{*}, 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 twofold. 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 structure using lattice operations was proposed in [11].

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].

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 $x \in \mathbb{R}^n$ 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 $x, y \in \mathbb{R}^n$, $x \vee y = (x_1 \vee y_1, \ldots, x_n \vee y_n)$ and $x \wedge y = (x_1 \wedge y_1, \ldots, x_n \wedge y_n)$, 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 $a \wedge -\infty = -\infty \wedge a = -\infty$. Setting $\mathbb{R}_{\infty} = \mathbb{R} \cup \{-\infty\}$ and $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{\infty\}$, we define $a + \infty = a + \infty + a = a + a = a + a = a + a + a = a + a + a + a = \cdots$ for all $a \in \mathbb{R}_{\infty}$, and $a + (\infty) = (\infty) + a = (\infty) + a = a + a = a + a = \cdots$ for all $a \in \mathbb{R}_{-\infty}$, where $+ = + \ast$ 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 $L_\infty$ metrics. The $L_1$ metric, defined as $d_1(x, y) = \sum_{i=1}^n |x_i - y_i|$, is known as the “city-block” or “Manhattan” distance and the $L_\infty$ metric, defined as $d_\infty(x, y) = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$, is known as the Chebyshev or “chessboard” distance.

The set $S_p^{n-1} = \{x \in \mathbb{R}^n : d_p(0, x) = 1\}$, where $0$ denotes the origin and $p \in [1, \infty]$, defines the $(n-1)$-
The geometry of $S_{n-1}^1$ 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 $x_1 + x_2 = -1, x_1 + x_3 = 1, x_1 - x_2 = -1, \text{and } x_1 - x_2 = 1$. Defining $E_1(x) = x_1 + x_2$ and $E_2(x) = x_1 - x_2$, it is easy to see that the set bounded by $S_1^1$ is given by

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

Similarly, the compact set bounded by the 2-sphere $S_2^2$ is the octahedron $P^3$ bounded by eight planes that are generated by the eight sets of affinely independent points:

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

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 $E_i(x) = x_1 + x_2 + x_3, E_2(x) = x_1 + x_2 - x_3, E_3(x) = x_1 - x_2 + x_3, \text{ and } E_4(x) = x_1 - x_2 - x_3$, it is not difficult to establish that

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

Equations 1 and 2 readily generalize to any dimension $n > 3$. Let $e^i, \ldots, e^n$ denote the standard orthonormal basis of the vector space $\mathbb{R}^n$, where $e^i$ is defined by $e^i_j = 1$ if $j = i$, and $e^i_j = 0$ if $j \neq i$. Observe that the vector $p^i = (1, 1) = (0, 0, 1, 1)$ is perpendicular to the two lines $E_1(x) = \pm 1$ in Eq. 1, while the vector $p^3 = (1, -1) = (1, 0, 0, -1)$ is orthogonal to the two planes $E_2(x) = \pm 1$ determined by $V_1$ and $V_4$, the vector $p^2 = (1, -1, 1) = (1, 0, 0, 1)$ is orthogonal to the two planes $E_3(x) = \pm 1$ determined by $V_2$ and $V_3$, the vector $p^3 = (1, 1, -1) = (1, 0, 0, 1)$ is orthogonal to the two planes $E_4(x) = \pm 1$ determined by $V_3$ and $V_4$, and the vector $p^4 = (1, -1, -1) = (1, 0, 0, 1)$ is orthogonal to the two planes $E_5(x) = \pm 1$ determined by $V_5$ and $V_6$. Next, let $\mathbb{Z}_{2^n} = \{ 0, 1, \ldots, 2^n - 1 \}$ denote the set of integers modulo $2^n$, while $\mathbb{Z}_2^n = \{ b : b = (b_1, \ldots, b_n), b_i \in \{ 0, 1 \}, i = 1, \ldots, n \}$ 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 $\beta(i-j)$ denote the bit value of the $j$-th coordinate of the binary number $b^i \in \mathbb{Z}_2^n$ representing the integer $i - 1 \in \mathbb{Z}_{2^n}$. That is,

$$\beta(i-j) = \text{mod}([i-j]/2^j), 2) = b^i_j \in [0, 1].$$

The bipolar vectors $p^i, \ldots, p^n$, 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, $p^1 = (1, -1, \ldots, (-1)^{i-1}, \ldots, (-1)^{n-1})$. If $E_i(x)$ is defined as the dot product $E_i(x) = p^i \cdot x$, then setting $E_i(x) = b_i$, where $b$ is an arbitrary constant, results in a hyperplane with $p^i$ orthogonal to this hyperplane. The $n$-dimensional polytope

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

where $E_i(x) = p^i \cdot x$ 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 $\{ w^i_1 : \ell \in [0, 1], w^i_1 < w^i_0, i = 1, \ldots, n \}$, then the set

$$H^n = \{ x \in \mathbb{R}^n : w^i_1 \leq x_i \leq w^i_0, i = 1, \ldots, n \}$$

is called an $n$-dimensional hyperbox. If for some non-empty subset $\{ i_1, \ldots, i_k \} \subset \{ 1, \ldots, n \}$ we have $w^i_j = w^i_0$ for $j = 1, \ldots, k$, then the set $\{ x \in \mathbb{R}^n : w^i_1 \leq x_i \leq w^i_0, i = 1, \ldots, n \}$ is an $(n-k)$-dimensional hyperbox in $\mathbb{R}^n$. If for some $i \in \{ 1, \ldots, n \}$, $w^i = -\infty$ or $w^i_0 = \infty$ (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 $\{ w^i_1 : \ell \in [0, 1], w^i_1 < w^i_0, i = 1, \ldots, 2^n - 1 \}$, then the set

$$P^n = \{ x \in \mathbb{R}^n : w^i_1 \leq E_i(x) \leq w^i_0, i = 1, \ldots, 2^n - 1 \}$$
is called a star polytope or $L_1$-polytope. Lower dimensional star polytopes and open endedness at $E_0(x)$ 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 $d_{\infty}$. 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 $d \in \{d_1, d_{\infty}\}$. 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 $\delta$ such that for any positive number $\varepsilon$ with $\varepsilon < \delta$ there exists an SLLP that assigns each point $x \in \mathbb{R}^n$ to class $C_j$ whenever $x \in X_j$ and not to class $C_i$ if $d(x, X_j) > \varepsilon$, where $j \in \{1, \ldots, m\}$. Furthermore, no point $x \in \mathbb{R}^n$ is assigned to more than one class.

The theorem and its proof are a straightforward generalization of the two-class theorem presented in [11]. Thus, a point $x \in \mathbb{R}^n$ within the $\varepsilon$-band surrounding $X_j$, but not in $X_j$, will be assigned to either class $C_j$ or $C_0 = (\bigcup_{j=1}^m C_j)^c$, 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 $k$th dendrite of $M_j$ receiving input from a terminal axonal branch of neuron $N_i$ is denoted by $w_{ijk}$, 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 $k$th 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 $k$th dendrite $d_{jk}$ of $M_j$ is given by

$$
\tau_{jk}(x) = p_{kj} \bigwedge_{i \in I_k, \ell \in L_k} (-1)^{1-\ell} (x_i + w_{ijk}^\ell),
$$

where $x = (x_1, \ldots, x_n)$ 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 $k$th dendrite of $M_j$, i.e., the set $\{N_i : i \in I_k\}$, $L_k \subseteq \{0,1\}$ corresponds to the set of terminal fibers of $N_i$ that synapse on the $k$th 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 $k$th dendrite of $M_j$ to the received input.

It follows from the formulation $L_{ik} \subseteq \{0,1\}$ that the $i$th 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 $k$th dendrite of $M_j$, then the incoming information $x \in \mathbb{R}^n$ will result in an excitatory post-synaptic response in the $k$th 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 $\tau_{kj}(x)$ 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

$$
\tau_j(x) = p_j \bigwedge_{k=1}^{K_j} \tau_{kj}(x),
$$
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 $y_j = f[\tau_j(x)]$. In this exposition we restrict our discussion to the hard-limiter

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

(9)

Victims 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 $R_1 = \{x \in \mathbb{R}^2 : 0 \leq x_i \leq 2, i = 1, 2\}$, 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.

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 $x \in \mathbb{R}^2$ is a point in the triangle if and only if $0 \leq x_i \leq 2$ for $i = 1, 2, 0 \leq E_1(x) \leq 4$, and $0 \leq E_2(x) \leq 2$. The corresponding lattice algebra expression satisfies the following inequality, $(2 - x_1) \land x_1 \land (2 - x_2) \land x_2 \land (4 - E_1(x)) \land E_1(x) \land (2 - E_2(x)) \land E_2(x) \geq 0$, and whose left side can be written as

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

(10)

where $w_0^0 = -2, w_1^0 = 0, w_0^1 = -2, w_1^1 = 0, \omega_1^0 = -4, \omega_1^1 = 0, \omega_2^0 = -2, \omega_2^1 = 0$, and $\tau(x) \geq 0$ if and only if $x \in H^2 \cap P^2$.

![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.](image)

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 $\tau_1(x) = \tau(x)$ for any input vector $x \in \mathbb{R}^2$ (see bottom of Fig. 4).

For an input vector $x$, a neuron $N_i$ transmits the value $x_i$ along its axonal branches to those synaptic sites on $d_1$ whose weight values are $w_i^{j1}$. In contrast, the neuron $E_i$ produces the output $E_i(x) = p_i^1 \cdot x$ and transmits it along its axon to $d_i$. The biological interpretation would be that $E_i$ transmits a spike train, with the spikes transmitting the variables making up the expression of $E_i(x)$. The terminal axonal fibers of $E_i$ impinge on the synaptic sites having weights $w_i^{j1}$.

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 $(d_1, d_{\infty})$-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 $(d_1, d_{\infty})$-model begins with a larger set of input neurons, namely $N_1, \ldots, N_n, E_1, \ldots, E_{n-1}$, 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 $N_j$ 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 $p_{k,j}^N$ and $p_{k,j}^E$, respectively. The total input received by the $k$th dendrite $d_{jk}$ of $M_j$ is given by

$$\tau_{k,j}(x) = p_{k,j}^N \bigwedge_{i \in I_k^N, \ell \in L_k^N} (-1)^{1-\ell}(x_i + w_{ij,k}^N) \land p_{k,j}^E \bigwedge_{i \in I_k^E, \ell \in L_k^E} (-1)^{1-\ell}(E_i(x) + w_{ij,k}^E),$$

where $p_{k,j}^N, p_{k,j}^E \in \{-1,1\}$, $I_k^N \subseteq \{1, \ldots, 2^n-1\}$ corresponds to the set of all input neurons $E_i$ with terminal fibers that synapse on the first branch of the $k$th dendrite of $M_j$, and $L_k^N, L_k^E$ have the same meaning as $I_k$ and $L_k$ in Eq. 7. The symbol $w_{ij,k}$ denotes the synaptic weight at synapse of $N_i$ on $d_{jk}^N$ and $w_{ij,k}$ 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 $p_{k,j}^N = p_{k,j}^E$, then $d_{jk}$ has no branching fibers. In this case we also allow the following additional simplification: If $w_{ij,k} = w_{ij,k}^N$ or $w_{ij,k} = w_{ij,k}^E$, 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 $\tau_j(x)$ as defined in Eq. 8. Note that, Eq. 11 remains simple in the sense that it only involves the lattice group operations of $(\mathbb{R}, \lor, \land, +)$ and no multiplications. As before, the value $\tau_{k,j}(x)$ is passed to the cell body of $M_j$. The total information computed by the dendrites is combined by the $M_j$, using the formulation $\tau_j(x) = p_j \bigwedge_{k=1}^{N_j} \tau_{k,j}(x)$ 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 $x = (x_1, x_2, x_3) \in \mathbb{Z}_2^3$ define its index $\xi$ by $\xi = 4x_1 + 2x_2 + x_3 + 1$, so that $x^1 = (0,0,0), x^2 = (0,0,1), \ldots, x^8 = (1,1,1)$. The class pattern of $x^k = (x_{k1}, x_{k2}, x_{k3})$, for $\xi = 1, \ldots, 8$, is defined in terms of the exclusive or operation $\oplus$ of its coordinates, $c_k = x_{k1} \oplus x_{k2} \oplus x_{k3}$. Specifically, class $C_1$ is given by $C_1 = \{x^k \in \mathbb{Z}_2^3 : c_k = 1\} = \{x^2, x^3, x^4, x^8\}$ while $C_0 = \{x^k \in \mathbb{Z}_2^3 : c_k = 0\} = \{x^1, x^4, x^6, x^7\}$. 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

$$\tau(x) = \bigwedge_{i \in I^N, \ell \in L^N} (-1)^{1-\ell}(E_i(x) + w_{ij,k}^N) \land (1 - E_j(x)) \land (1 - E_k(x)) \land (E_k(x) + 1),$$

where $L_j^F = L_k^F = \{1\}$, $L_j^E = L_k^E = \{0\}$, $\omega_1 = \omega_2 = \omega_3 = -1$, and $\omega_4 = 1$. 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].
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 \( L_\infty \) 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 \( T = \{x^j \in \mathbb{R}^n : \xi = 1, \ldots, t\} \) denote the training set for an \( m \) class problem \( C_1, \ldots, C_m, P = \{1, \ldots, t\} \), \( T_j = \{x^j \in T : x^j \in C_j\} \) for \( j = 1, \ldots, m \), and \( c_j^t \in \{0, 1\} \), where \( c_j^t = 1 \) if \( x^j \in T_j \) and \( c_j^t = 0 \) if \( x^j \notin T_j \). Also, \( T_j = T \setminus T_j \). The sets \( I_k^N, L_k^N, I_k^E, L_k^E \) appearing in Eq. 11 are generated during training.

The following values are pre-established: for \( \ell = 0, 1 \) and \( j = 1, \ldots, m \) compute \( \varepsilon_j^{\mathcal{N}} = \bigwedge_{y \in T_j} d_\infty(x, y) \), and \( \delta_j^{\mathcal{IN}} = (1 - \varepsilon_j^{\mathcal{N}}) \). Similarly, \( \varepsilon_j^{\mathcal{E}} = \bigwedge_{y \in T_j} d_1(x, y) \), where \( d_1 \) is the \( L_1 \) metric.

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 \( k = 1 \); \( I_k^N = \{1, \ldots, n\} \); \( I_k^E = \{1, \ldots, 2^{n-1}\} \)

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

**S2** for \( i \in I_k^N \)

\[
w_{ijk}^{0,N} = \left\lfloor \bigvee_{c_j = 1} x^j \right\rfloor + \delta_j^{0,N} \]

\[
w_{ijk}^{1,N} = \left\lfloor \bigwedge_{c_j = 1} x^j \right\rfloor + \delta_j^{1,N} ; \quad L_k^N = \{0, 1\} \]

for \( i \in I_k^E \)

\[
\omega_{ijk}^{0,E} = \left\lfloor \bigvee_{c_j = 1} E_i(x^j) \right\rfloor + \delta_j^{0,E} \]

\[
\omega_{ijk}^{1,E} = \left\lfloor \bigwedge_{c_j = 1} E_i(x^j) \right\rfloor + \delta_j^{1,E} ; \quad L_k^E = \{0, 1\} \]

[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** \( p_{ijk}^{0,N} = p_{ijk}^{1,E} = (1 - \varepsilon_j^{\mathcal{N}}) \)

for \( \xi = 1 \) to \( t \)

\[
\tau_{ijk}^{0,N}(x^j) = p_{ijk}^{0,N} \bigwedge_{i \in I_k^E} \bigwedge_{c_j \in L_k^E} (1 - \varepsilon_j^{\mathcal{E}}) + w_{ijk}^{0,E} \]

\[
\tau_{ijk}^{1,E}(x^j) = p_{ijk}^{1,E} \bigwedge_{i \in I_k^E} \bigwedge_{c_j \in L_k^E} (1 - \varepsilon_j^{\mathcal{E}}) + w_{ijk}^{1,E} \]

\[
\tau_{ijk}(x^j) = \tau_{ijk}^{0,N}(x^j) \land \tau_{ijk}^{1,E}(x^j) \]

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

**S4** for \( \xi = 1 \) to \( t \)

\[
h_k = \{x^t \in T : f(\tau_j(x^t)) = 1\} \]

if \( R_k \cap T_j = \emptyset \) or, if \( f(\tau_j(x^t)) = c_j^t, \forall \xi \in \mathcal{P} \)

\[
K_j = k \quad \text{and return to S1} \]

else randomly choose \( x^t \in R_k \cap T_j \)

[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 \( x^t \)]

**S5** \( k = k + 1 \)

\[
I_k^N = I_k^0 \cup I_k^1 \quad \text{for} \quad i \in I_k^N, \quad L_k = L_k^N \}

\[
I_k^E = I_k^1 \quad \text{for} \quad i \in I_k^E, \quad L_k = L_k^E \}

[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 \( x^t \in T_j \)

if \( x^t > x^j \), then \( p = p + 1, s_p = \xi \), else continue

if \( x^t < x^j \), then \( q = q + 1, t_q = \xi \), else continue

if \( p > 0 \)

\[
d = \bigwedge_{c_j = 1} d_\infty(y, x^t) ; \quad w_{ijk}^{0,E} = -(x^t - \frac{1}{2} d) \]

else \( I_k^0 \cup I_k^1 \quad \text{for} \quad i \in I_k^E, \quad L_k = L_k^E \}

if \( q > 0 \)

\[
d = \bigwedge_{c_j = 1} d_\infty(y, x^t) ; \quad w_{ijk}^{1,E} = -(x^t + \frac{1}{2} d) \]

else \( I_k^0 \cup I_k^1 \quad \text{for} \quad i \in I_k^E, \quad L_k = L_k^E \}

[Cycle through input neurons of type \( E \) and assign \( L_\infty \) metric weights to new synaptic sites that may classify \( x^t \) 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 \( x^t \in T_j \)

if \( E_i(x^t) > E_i(x^j) \), then \( p = p + 1, s_p = \xi \), else continue

if \( E_i(x^t) < E_i(x^j) \), then \( q = q + 1, t_q = \xi \), else continue

if \( p > 0 \)

\[
d = \bigwedge_{c_j = 1} d_1(y, x^t) ; \quad w_{ijk}^{0,E} = -(E_i(x^t) + \frac{1}{2} d) \]

else \( I_k^0 \cup I_k^1 \quad \text{for} \quad i \in I_k^E, \quad L_k = L_k^E \}

if \( q > 0 \)

\[
d = \bigwedge_{c_j = 1} d_1(y, x^t) ; \quad w_{ijk}^{1,E} = -(E_i(x^t) - \frac{1}{2} d) \]

else \( I_k^0 \cup I_k^1 \quad \text{for} \quad i \in I_k^E, \quad L_k = L_k^E \}

[Cycle through input neurons of type \( E \) and assign \( L_1 \) metric weights to new synaptic sites that may classify \( x^t \) 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 priori, probabilistic, or statistical knowledge derived from the data or training set under consideration. Thus, e.g., the parameters $\delta_i^j$ associated to neurons of type $N$ or $E$, are independent of the coordinate index $i$ but could be generalized to $\delta_{ij}$ by including knowledge obtained from the spatial distribution of the $ij$th coordinates $x^j_i$ 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 $B^n = \{x \in \mathbb{R}^n : x_i \in \mathbb{Z}_2, i = 1, \ldots, n\}$. 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 $T_{jp}^c = X_j \setminus T_{jp}$. 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, $\tau$ is the number of runs, $\rho_p$ is the number of correctly classified test patterns, and $\mu_p$ denotes the number of misclassified test patterns in each run, then the average fraction of hits is given by,

$$f_p = 1 - \frac{\hat{\mu}}{k} ; \hat{\mu} = \frac{1}{\tau} \sum_{r=1}^{\tau} \mu_r ; \hat{\rho} = \frac{1}{\tau} \sum_{r=1}^{\tau} \rho_r , \quad (13)$$

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 $\alpha^N_j = \alpha^E_j = 0.49$ 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 $L_{\infty}$ and $L_1$ lattice metrics, the 3rd column gives the average number of misclassified inputs, the 4th and following columns show the average number of dendrites per class, $K_j$ 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 $X \subset \mathbb{R}^2$. Points belonging to class $C_1$ are marked by a circle (○) and points of class $C_2$ are marked with a disk (●).](image)

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\rho$</th>
<th>$\hat{\rho}$</th>
<th>$K_1$</th>
<th>$K_2$</th>
<th>$f_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>241</td>
<td>53</td>
<td>3</td>
<td>2</td>
<td>0.820</td>
</tr>
<tr>
<td>20%</td>
<td>263</td>
<td>31</td>
<td>9</td>
<td>6</td>
<td>0.895</td>
</tr>
<tr>
<td>22%</td>
<td>294</td>
<td>0</td>
<td>21</td>
<td>21</td>
<td>1.000</td>
</tr>
<tr>
<td>30%</td>
<td>267</td>
<td>27</td>
<td>17</td>
<td>7</td>
<td>0.908</td>
</tr>
<tr>
<td>40%</td>
<td>278</td>
<td>16</td>
<td>18</td>
<td>11</td>
<td>0.945</td>
</tr>
<tr>
<td>50%</td>
<td>279</td>
<td>15</td>
<td>20</td>
<td>18</td>
<td>0.949</td>
</tr>
<tr>
<td>60%</td>
<td>283</td>
<td>11</td>
<td>14</td>
<td>25</td>
<td>0.963</td>
</tr>
<tr>
<td>70%</td>
<td>286</td>
<td>8</td>
<td>15</td>
<td>34</td>
<td>0.973</td>
</tr>
<tr>
<td>80%</td>
<td>288</td>
<td>6</td>
<td>34</td>
<td>32</td>
<td>0.980</td>
</tr>
<tr>
<td>90%</td>
<td>291</td>
<td>3</td>
<td>43</td>
<td>41</td>
<td>0.990</td>
</tr>
</tbody>
</table>

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.
Note that for the data sets ‘Twins’ and ‘Iris’, a high average fraction of hits such as \( f_p > 0.95 \) is obtained for percentages \( p \) as low as 60%. Furthermore, the entry in Table II marked with a 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, \( f_{50} = 0.953 \) and \( f_{50} = 0.973 \) (with \( \tau = 2 \)), respectively, or in comparison with an Edge-effect Fuzzy Support Vector Machine [28] whose \( f_{60} = 0.978 \) (here, \( \tau \) is not specified explicitly).

### V. CONCLUSIONS

This paper introduces an extended training algorithm for neural networks endowed with dendrites based on the lattice metrics \( L_\infty \) 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


