An Introduction to Morphological Neural Networks

Gerhard X. Ritter (ritter@cis.ufl.edu) and Peter Sussner (ps0@cis.ufl.edu)
University of Florida
Center for Computer Vision and Visualization
Room E301, CSE Building
Gainesville, Florida 32611

Abstract

The theory of artificial neural networks has been successfully applied to a wide variety of pattern recognition problems. In this theory, the first step in computing the next state of a neuron or in performing the next layer neural network computation involves the linear operation of multiplying neural values by their synaptic strengths and adding the results. Thresholding usually follows the linear operation in order to provide for nonlinearity of the network. In this paper we introduce a novel class of neural networks, called morphological neural networks, in which the operations of multiplication and addition are replaced by addition and maximum (or minimum), respectively. By taking the maximum (or minimum) of sums instead of the sum of products, morphological network computation is nonlinear before thresholding. As a consequence, the properties of morphological neural networks are drastically different than those of traditional neural network models. In this paper we consider some of these differences and examine the computing capabilities of morphological neural networks. As particular examples of a morphological neural network we discuss morphological associative memories and morphological perceptrons.

1. Introduction

The concept of morphological neural networks grew out of the theory of image algebra [22, 17, 16, 23]. It was shown that a subalgebra of image algebra includes the mathematical formulations of currently popular neural network models [20, 16] and first attempts in formulating useful morphological neural networks appeared in [18, 10]. Since then, only a few papers involving morphological neural networks have appeared. J. L. Davidson employed morphological neural networks in order to solve template identification and target classification problems [9, 12, 11, 8]. C.P. Suarez-Araujo applied morphological neural networks to compute homothetic auditory and visual invariances [27, 26]. Both of these researchers devised multilayer morphological neural networks for very specialized applications. In this paper we attempt a more general approach to morphological neural networks which, hopefully, will lay the foundations for future research efforts concerned with the behavior, capabilities, and applications of these novel networks.

We need to remark that an entirely different model of a morphological network was presented in [28]. This particular model uses the usual operations of multiplication and summation at each node, which is fundamentally different from the models presented here. The model presented here is connected with the fundamental question concerning the difference between biological neural networks and artificial neural networks: Is the strength of the electric potential of a signal traveling along an axon the result of a multiplicative process and does the mechanism of the postsynaptic membrane of a neuron add the various potentials of electrical impulses, or is the strength of the electric potential an additive process and does the postsynaptic membrane only accept signals of a certain maximum strength? A positive answer to the latter query would provide a strong biological basis for morphological neural networks.

2. Computational Basis for Morphological Neural Networks

In recent years lattice based matrix operations have found widespread applications in the engineering sciences. In these applications, the usual matrix operations of addition and multiplication are replaced by corresponding lattice operations. Lattice induced matrix operations lead to an entirely different perspective of a class of nonlinear transformations. These ideas were applied by Shimbel [25] to communications networks, and to machine scheduling by Cuninghame-Green [4, 5] and Giffler [13]. Others have discussed their usefulness in applications to shortest path problems in graphs [15, 2, 3, 1]. Additional examples are given in [6], primarily in the field of operations research. Application to image processing were first developed by Ritter and Davidson [19, 7]. This paper presents a continuation of these developments in that we take lattice based operations as the basic computational model for artificial neural networks.

Artificial neural network models are specified by the network topology, node characteristics, and training or learning rules. The underlying algebraic system used in these models is the set of real numbers $\mathbb{R}$ together with the operations of addition and multiplication and the laws governing these operations. This algebraic system, known as a *ring*, is commonly denoted by $(\mathbb{R},+,\times)$. The two basic equations governing the theory of computation in the standard neural network model are:

$$\tau_i(t+1) = \sum_{j=1}^{n} a_j(t)\cdot w_{ij} \tag{1}$$

and

$$a_i(t+1) = f(\tau_i(t+1) - \theta_i), \tag{2}$$

where $a_j(t)$ denotes the value of the *j*th neuron at time *t*, n represents the number of neurons in the network, w_{ij} the synaptic connectivity value between the *i*th neuron and the *j*th neuron, $\tau_i(t+1)$ the next total input effect on the *i*th neuron, θ_i a threshold, and f the next state function which usually introduces a nonlinearity into the network. Although not all current network models can be precisely described by these two equations, they nevertheless can be viewed as variations of these.

Note that the computations represented by Equation 1 are based on the operations of the algebraic structure $(\mathbb{R},+,\times)$. The basic computations occurring in the proposed morphological network are based on the *semi-ring* structures $(\mathbb{R}_{-\infty},\vee,+)$ and $(\mathbb{R}_{\infty},\wedge,+')$, where $\mathbb{R}_{-\infty}$ and $\mathbb{R}_{\infty}$ represent the extended real number systems $\mathbb{R}_{-\infty} = \mathbb{R}\cup\{-\infty\}$ and $\mathbb{R}_{\infty} = \mathbb{R}\cup\{\infty\}$. The basic arithmetic and logic operations in the extended real number systems are as follows. The symbol $+$ denotes the usual addition with the additional stipulation that $a+(-\infty) = (-\infty)+a = -\infty \ \forall a \in \mathbb{R}_{-\infty}$. The symbol $+'$ denotes the *self dual* of addition and is defined by $a+'b \equiv a+b$ whenever $a,b\in\mathbb{R}$, and $a+'\infty = \infty+'a = \infty \ \forall a \in \mathbb{R}_{\infty}$. The symbols $\vee$ and $\wedge$ denote the binary operations of maximum and minimum, respectively, with the additional stipulation that $a\vee(-\infty) = (-\infty)\vee a = a \ \forall a \in \mathbb{R}_{-\infty}$ and $a\wedge\infty = \infty\wedge a = a \ \forall a\in\mathbb{R}_{\infty}$. Note that the symbol $-\infty$ acts like a *zero* element in the system $(\mathbb{R}_{-\infty},\vee,+)$ if one views $\vee$ as *addition* and $+$ as *multiplication*. Similar comments hold for the symbol ∞ in the system $(\mathbb{R}_{\infty},\wedge,+')$. Also, the role of the multiplicative identity in the structure $(\mathbb{R},+,\times)$ is played by the number 1; i.e., $1\cdot a = a\cdot 1 = a \quad \forall a\in\mathbb{R}$. In the structures $(\mathbb{R}_{-\infty},\vee,+)$ and $(\mathbb{R}_{\infty},\wedge,+')$, this role is played by the number 0 since $0+a = a+0 = a \quad \forall a \in \mathbb{R}$.

Using the structure $(\mathbb{R}_{-\infty},\vee,+)$, the two basic equations underlying the theory of computation in the morphological neural network model are given by:

$$\tau_i(t+1) = \bigvee_{j=1}^{n} a_j(t) + w_{ij} \tag{3}$$

and

$$a_i(t+1) = f(\tau_i(t+1) - \theta_i). \tag{4}$$

Observe that Equations 2 and 4 are identical. Thus, the difference between the classical models and the morphological model is the computation of the next total input effect on the *i*th neuron, which is given by

$$\bigvee_{j=1}^{n} a_j(t) + w_{ij}$$
$$= (a_1(t)+w_{i1})\vee(a_2(t)+w_{i2})\vee\cdots\vee(a_n(t)+w_{in}). \tag{5}$$

Using the dual structure $(\mathbb{R}_{\infty},\wedge,+')$ instead, then Equation 3 needs to be replaced by

$$\tau_i(t+1) = \bigwedge_{j=1}^{n} a_j(t) +' w_{ij}. \tag{6}$$

Equations 2 and 4 represent the basic operations of a *dilation* and an *erosion* that form the foundation of mathematical morphology [24]. Hence the reason for calling the proposed neural network model a *morphological* neural network.

Let $\mathbf{v}'$ denote the transpose of the vector $\mathbf{v}$. The total network computation resulting from Equation 1 can be expressed in matrix form as

$$\mathrm{T}(t+1) = W\cdot \mathbf{a}(t), \tag{7}$$

where W denotes the $n\times n$ synaptic weight matrix whose *i,j*th entry is w_{ij} and $\mathbf{a}(t) = (a_1(t),\ldots,a_n(t))'$, $\mathrm{T}(t+1) = (\tau_1(t+1),\ldots,\tau_n(t+1))'$.

Analogous to Equation 7, the total morphological network computation resulting from Equation 3 (or Equation 6) can also be expressed in matrix form. In order to do this, we need to define a matrix product in terms of the operations of the semi-ring $(\mathbb{R}_{-\infty},\vee,+)$. For an $m\times p$ matrix A and a $p\times n$ matrix B with entries from $\mathbb{R}_{-\infty}$, the matrix *product* $C = A \boxed{\vee} B$, also called the *max product* of A and B, is defined by

$$c_{ij} = \bigvee_{k=1}^{p} a_{ik} + b_{kj}$$
$$= (a_{i1}+b_{k1})\vee(a_{i2}+b_{2j})\vee\ldots\vee(a_{ip}+b_{pj}). \tag{8}$$

The *min product* of A and B induced by $(\mathbb{R}_{\infty},\wedge,+')$ is defined in a similar fashion. Specifically, the *i,j*th entry of $C = A \boxed{\wedge} B$ is given by

$$c_{ij} = \bigwedge_{k=1}^{n} a_{ik} +' b_{kj}$$
$$= (a_{i1}+'b_{k1})\wedge(a_{i2}+'b_{2j})\wedge\ldots\wedge(a_{ip}+'b_{pj}). \tag{9}$$

The total network computation resulting from Equations 3 and 6 can now be expressed in the matrix forms

$$\mathrm{T}(t+1) = W \boxed{\vee}\, \mathbf{a}(t) \tag{10}$$

and

$$\mathrm{T}(t+1) = W \boxed{\wedge}\, \mathbf{a}(t), \tag{11}$$

respectively.

Some additional comments concerning lattice based operations are pertinent when discussing morphological network computations. When using the lattice $(\mathbb{R}_{-\infty}, \vee, +)$, the maximum of two matrices replaces the usual matrix addition of linear algebra. Here the *i,j*th entry of the matrix $C = A \vee B$ is given by $c_{ij} = a_{ij} \vee b_{ij}$. Similarly, the minimum of two matrices $C = A \wedge B$, which is used when employing the structure $(\mathbb{R}_{\infty}, \wedge, +')$, is defined by $c_{ij} = a_{ij} \wedge b_{ij}$.

The structures $(\mathbb{R}_{-\infty}, \vee, +)$ and $(\mathbb{R}_{\infty}, \wedge, +')$ can be combined into one cohesive algebraic system $(\mathbb{R}_{\pm\infty}, \vee, \wedge, +, +')$, called a *bounded l-group*. Here $\mathbb{R}_{\pm\infty} = \mathbb{R} \cup \{+\infty, \infty\}$ and the operations of maximum and minimum extend intuitively to the appended symbols of infinity, namely

$$\begin{aligned} \infty \vee r = r \vee \infty = \infty \quad & \forall r \in \mathbb{R}_{\pm\infty} \\ \infty \wedge r = r \wedge \infty = r \quad & \forall r \in \mathbb{R}_{\pm\infty} \end{aligned} \tag{12}$$

The dual additive operations $+$ and $+'$ are identical when adding real numbers; i.e.,

$$a +' b = a + b \ \ \forall\, a, b \in \mathbb{R}. \tag{13}$$

However, they introduce an asymmetry between $-\infty$ and $+\infty$ when properly extended to the set $\mathbb{R}_{\pm\infty}$. Specifically, we define

$$\begin{aligned} & a + (-\infty) = (-\infty) + a = -\infty && a \in \mathbb{R}_{-\infty} \\ & a + \infty = \infty + a = \infty && a \in \mathbb{R}_{\infty} \\ & a +' (-\infty) = (-\infty) +' a = -\infty && a \in \mathbb{R}_{-\infty} \\ & a +' \infty = \infty +' a = \infty && a \in \mathbb{R}_{\infty} \\ & (-\infty) + \infty = \infty + (-\infty) = -\infty && \\ & (-\infty) +' \infty = \infty +' (-\infty) = \infty && \end{aligned} \tag{14}$$

The algebraic structure of $\mathbb{R}_{\pm\infty}$ provides for an elegant duality between matrix operations. If $r \in \mathbb{R}_{\pm\infty}$, then the *additive conjugate* of r is the unique element r^* defined by

$$r^* \equiv \begin{cases} -r & \text{if } r \in \mathbb{R} \\ -\infty & \text{if } r = +\infty . \\ +\infty & \text{if } r = -\infty \end{cases} \tag{15}$$

Here, $-r$ is the inverse of r under the group operation $+$. Therefore, $(r^*)^* = r$. This gives the following relation for all r, u in $\mathbb{R}_{\pm\infty}$:

$$r \wedge u = (r^* \vee u^*)^*. \tag{16}$$

If $A = (a_{ij})_{m \times n}$ is an $m \times n$ matrix with $a_{ij} \in \mathbb{R}_{\pm\infty}$, then the *conjugate matrix* A^* of A is the $n \times m$ matrix $A^* = (b_{ij})_{n \times m}$ defined by $b_{ij} = [a_{ji}]^*$, where $[a_{ji}]^*$ is the additive conjugate of a_{ji} as defined above. It follows that

$$A \wedge B = (A^* \vee B^*)^* \tag{17}$$

and

$$A \boxed{\wedge} B = (B^* \boxed{\vee} A^*)^* \tag{18}$$

for appropriately sized matrices. This implies that a morphological neural net using the operation $\boxed{\vee}$ (i.e., Equation 10) can always be reformulated in terms of the operation $\boxed{\wedge}$ (i.e., Equation 11), and vice versa, by using the duality relations expressed in Equations 17 and 18.

Having defined the necessary mathematical tools, we are now in a position to discuss some basic properties of morphological neural networks and present some examples.

3. Computing Capabilities of Morphological Neural Networks

In this section we show that a morphological neural network is formally capable of solving any conventional computational problem. Conventional computational problems are a class of problems whose inputs and outputs can be represented as binary strings. Given a problem whose input and output can be represented as binary strings $\mathbf{x} = x_1 x_2 \cdots x_m$ and $\mathbf{y} = y_1 y_2 \cdots y_k$, respectively, then the problem can be characterized in terms of k Boolean functions

$$y_1 = f_1(\mathbf{x}),\ y_2 = f_2(\mathbf{x}),\ \ldots,\ y_k = f_k(\mathbf{x}) . \tag{19}$$

A well-known fact of switching theory is that two-input NAND gates form a complete basis for Boolean functions [14]. Recall that a two-input NAND gate is one whose output is 0 if and only if both inputs are 1. Being a complete basis means that any Boolean function can be simulated by a combinatorial circuit all of whose gates are exclusively two-input NAND gates.

In order to prove that a morphological neural network is formally capable of solving any conventional computational problem, we simply construct a morphological neural network which is equivalent to a combinatorial circuit all of whose gates are exclusively two-input NAND gates. The network is based on Equations 6 and 4 using the structure $(\mathbb{R}_{\infty}, \wedge, +')$. By the duality criteria, we could just as well use Equations 3 and 4 with the structure $(\mathbb{R}_{-\infty}, \vee, +)$. The morphological network which will simulate any such combinatorial circuit obeys the following computational rules:

1. $\tau_i(t+1) = \bigwedge_{j=1}^{n} (a_j(t) +' w_{ij})$

2. The value of the i-th neuron at time t is given by $f(\tau_i(t+1))$. Hence,

$$a_i(t+1) = \begin{cases} 0 & if\ \tau_i(t+1) > \theta_i \\ a_i(t) & if\ \tau_i(t+1) = \theta_i \\ 1 & if\ \tau_i(t+1) < \theta_i \end{cases} \tag{20}$$

3. The neurons a_i fire at random, one at each discrete time step.

These computational rules are applied at a neuron when computing the next state, except at the input neurons. At the input neurons, the function f is simply the identity function $f(x) = x$ so that $a_i(t+1) = \tau_i(t+1)$. The morphological network that will simulate a single NAND gate is shown in Figure 1. Here the neurons a_1 and a_2 receive input and the input gets moved (replicated) into a_3 and a_4. The initial state of a_5 is arbitrary (i.e., simply assign either state 0 or state 1 to a_5). Thus, the initial conditions are

$$\begin{aligned} a_1(0) = a_3(0) = x_1 \\ a_2(0) = a_4(0) = x_2 \end{aligned} \tag{21}$$

and $a_5(0) \in \{0, 1\}$ arbitrary chosen. The zero weights between a_1 and a_3, and between a_2 and a_4, respectively, serve as a buffer and prevents any parasitic feedback should one of these neurons fire at random. In other words, the states of the neurons a_1 and a_2 will not be affected, no matter which neuron in the network examines itself first. The threshold $\theta = 1$ is applied at neuron a_5. Neural connections that are not shown are assumed to have infinite weight ∞. Since we are taking minimums, they do not affect the computation. The output of this network is the state of the neuron a_5 after the net has stabilized. It is not difficult to see that this network simulates a NAND gate.

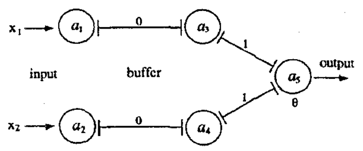

Fig. 1. A morphological net simulation a NAND gate.

To obtain a morphological network that simulates a combinatorial circuit of NAND gates, we simply replace each NAND gate with the neural network equivalent shown in Figure 2. For each input x_i, we provide an input neuron a_i and an associated second neuron a_j with a buffer memory $w_{ij} = 0 = w_{ji}$ between these two neurons as shown in Figure 2. To prevent parasitic feedback in the net, we simply increase weights and thresholds by one each time we go from one gate to the *next* gate in the cascade of the combinatorial circuit. Under these conditions, the *next* neuron (=gate) will not be able to effect the previous gate through its synaptic connection. Figure 2 illustrates this situation for three inputs. If as indicated in the figure, a fourth input is added, then the next output neuron would have threshold $\theta_3 = 3$ and the weights connecting to that output neuron would have all have the same value $w = 3$.

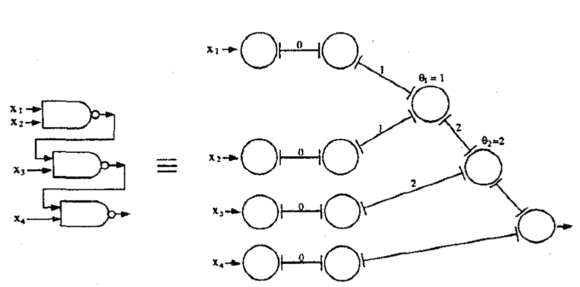

Fig. 2. A morphological net simulation of a three NAND gate combinatorial circuit.

This brute force approach only proves that any conventional type of computational problem can be done using morphological neural nets. It does not consider issues such as efficiency and peculiarities of morphological nets. Obviously, computations at a neuron are generally more efficient than those using classical models since only additions and logic operations (max or mins) are involved. Also, in the models that we have examined, which include morphological analogues of Hopfield nets, perceptrons, and feed-forward nets with learning rules, convergence is usually instantaneous or far more rapid than in their classical counterparts.

4. Morphological Associative Memories

In classical neural network theory, for a given input vector or *key* $\mathbf{x} = (x_1, \ldots, x_n)'$, an *associative memory* W recalls a vector output signal $f(\mathbf{y})$, where $\mathbf{y} = W \cdot \mathbf{x}$. If $f(y_i) = y_i\ \forall i$, then $f(\mathbf{y}) = \mathbf{y}$ and the memory is called a *linear associative memory*. A basic question concerning associative memories is: What is the simplest way to store k vector pairs $(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^k, \mathbf{y}^k)$, where $\mathbf{x}^\xi \in \mathbb{R}^n$

and $\mathbf{y}^\xi \in \mathbb{R}^m$ for $\xi = 1, \ldots, k$, in an $m \times n$ memory W? The well-known answer to that question is to set

$$W = \sum_{\xi=1}^{k} \mathbf{y}^\xi \cdot (\mathbf{x}^\xi)'. \tag{22}$$

In this case, the *i,j*th entry of W is given by $w_{ij} = \sum_{\xi=1}^{k} y_i^\xi x_j^\xi$. If the input patterns $\mathbf{x}^1, \ldots, \mathbf{x}^k$ are orthonormal, that is

$$(\mathbf{x}^j)' \cdot \mathbf{x}^i = \begin{cases} 1 & if\ i = j \\ 0 & if\ i \neq j \end{cases}, \tag{23}$$

then

$$W \cdot \mathbf{x}^i = \left(\mathbf{y}^1 \cdot (\mathbf{x}^1)' + \cdots + \mathbf{y}^k \cdot (\mathbf{x}^k)'\right) \cdot \mathbf{x}^i = \mathbf{y}^i. \tag{24}$$

Thus, we have *perfect recall* of the output patterns $\mathbf{y}^1, \ldots, \mathbf{y}^k$. If $\mathbf{x}^1, \ldots, \mathbf{x}^k$ are not orthonormal (which they are not in most realistic cases), then the term

$$N = \sum_{i \neq j} \mathbf{y}^j \cdot \left((\mathbf{x}^j)' \cdot \mathbf{x}^i\right) \neq 0 \tag{25}$$

is called the *noise term* and contributes to *cross talk* to the recalled pattern by additively modulating the signal term. Therefore filtering processes using threshold functions become necessary in order to retrieve the desired output pattern.

Morphological associative memories, which are based on the lattice algebra described in the preceding section, are surprisingly similar to these classical associative memories. Suppose we are given a vector pair $\mathbf{x} = (x_1, \ldots, x_n)' \in \mathbb{R}^n$ and $\mathbf{y} = (y_1, \ldots, y_m)' \in \mathbb{R}^m$. An associative morphological memory that will recall the vector $\mathbf{y}$ when presented the vector $\mathbf{x}$ is given by

$$W = \mathbf{y} \boxtimes (-\mathbf{x})' = \begin{pmatrix} y_1 - x_1 & \cdots & y_1 - x_n \\ \vdots & \ddots & \vdots \\ y_m - x_1 & \cdots & y_m - x_n \end{pmatrix} \tag{26}$$

since W satisfies the equation $W \boxtimes \mathbf{x} = \mathbf{y}$ as can be verified by the simple computation

$$W \boxtimes \mathbf{x} = \begin{pmatrix} \bigvee_{i=1}^{n} (y_1 - x_i + x_i) \\ \vdots \\ \bigvee_{i=1}^{n} (y_m - x_i + x_i) \end{pmatrix} = \mathbf{y}. \tag{27}$$

Note the similarity between Equation 22 and Equation 26 when $k = 1$. In the linear domain, $\mathbf{x}' \cdot \mathbf{x} = 1$ for a normalized vector. Equation 26 also incorporates a type of normalization for the pattern $\mathbf{x}$ in the sense that $(-\mathbf{x})' \boxtimes \mathbf{x} = 0$, which represents the *multiplicative* identity in the system $(\mathbb{R}_{-\infty}, \vee, +)$. The natural question one may ask is as to whether or not this concept can be extended to cases where $k > 1$. The answer is a qualified yes due to the fact that problems which in some way are analogous to those associated with ordinary associative memories (i.e., Equation 25) also occur in morphological associative memories.

Henceforth, let $(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^k, \mathbf{y}^k)$ be k vector pairs with $\mathbf{x}^\xi = \left(x_1^\xi, \ldots, x_n^\xi\right)' \in \mathbb{R}^n$ and $\mathbf{y}^\xi = \left(y_1^\xi, \ldots, y_m^\xi\right)' \in \mathbb{R}^m$ for $\xi = 1, \ldots, k$. The *optimal* morphological associative memory for recalling the vector $\mathbf{y}^\xi$ when presented with the pattern $\mathbf{x}^\xi$ for $\xi = 1, \ldots, k$ is given by

$$W = \bigwedge_{\xi=1}^{k} \left(\mathbf{y}^\xi \boxtimes (-\mathbf{x}^\xi)'\right). \tag{28}$$

Before examining the claim of optimality a bit closer, consider the following example.

Example 1. Let

$$\mathbf{x}^1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{y}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},$$
$$\mathbf{x}^2 = \begin{pmatrix} 0 \\ -2 \\ -4 \end{pmatrix}, \quad \mathbf{y}^2 = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}, \tag{29}$$
$$\mathbf{x}^3 = \begin{pmatrix} 0 \\ -3 \\ 0 \end{pmatrix}, \quad \mathbf{y}^3 = \begin{pmatrix} 0 \\ -2 \\ 0 \end{pmatrix}.$$

According to Equation 28, the memory W is given by

$$W = \bigwedge_{\xi=1}^{k} \left(\mathbf{y}^\xi \boxtimes (-\mathbf{x}^\xi)'\right)$$
$$= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \wedge \begin{pmatrix} -1 & 1 & 3 \\ -1 & 1 & 3 \\ 0 & 2 & 4 \end{pmatrix} \wedge \begin{pmatrix} 0 & 3 & 0 \\ -2 & 1 & -2 \\ 0 & 3 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} -1 & 0 & 0 \\ -2 & 1 & -2 \\ 0 & 0 & 0 \end{pmatrix}. \tag{30}$$

It can be easily verified that $W \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi$ holds for $\xi = 1, 2, 3$. For example,

$$W \boxtimes \mathbf{x}^1 = \begin{pmatrix} -1 & 0 & 0 \\ -2 & 1 & -2 \\ 0 & 0 & 0 \end{pmatrix} \boxtimes \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \mathbf{y}^1. \tag{31}$$

For the remainder of this section let W denote the memory given by Equation 28. For $\xi = 1, \ldots, k$, let $\mathbf{z}^\xi = \left(z_1^\xi, \ldots, z_m^\xi\right)'$ denote the output pattern when W is presented with the input pattern $\mathbf{x}^\xi$ (i.e., $W \boxtimes \mathbf{x}^\xi = \mathbf{z}^\xi$). Thus, if $i \in \{1, \ldots, m\}$ is an arbitrary index, then

$$\begin{aligned} z_i^\xi &= \bigvee_{j=1}^{n} \left(w_{ij} + x_j^\xi\right) \\ &= \bigvee_{j=1}^{n} \left(\bigwedge_{\zeta=1}^{k} \left[y_i^\zeta - x_j^\zeta\right] + x_j^\xi\right) \\ &\leq \bigvee_{j=1}^{n} \left(\left[y_i^\xi - x_j^\xi\right] + x_j^\xi\right) \\ &= y_i^\xi . \end{aligned} \tag{32}$$

Since i was arbitrary, we have that $\mathbf{z}^\xi \leq \mathbf{y}^\xi$ (i.e., $z_i^\xi \leq y_i^\xi \;\; \forall i = 1, \ldots, m$). This shows that

$$W \boxtimes \mathbf{x}^\xi \leq \mathbf{y}^\xi \;\; \forall \xi = 1, \ldots, k . \tag{33}$$

Now if $A = (a_{ij})_{m \times n}$ is a memory with the property that $A \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi$ for $\xi = 1, \ldots, k$, then for any arbitrary index $i \in \{1, \ldots, m\}$ the following equalities hold:

$$\bigvee_{j=1}^{n} \left(a_{ij} + x_j^\xi\right) = y_i^\xi \;\; \forall \xi = 1, \ldots, k . \tag{34}$$

It follows that for an arbitrary index $j \in \{1, \ldots, n\}$ we have

$$\begin{aligned} & a_{ij} + x_j^\xi \leq y_i^\xi \;\; \forall \xi = 1, \ldots, k \\ \Leftrightarrow\; & a_{ij} \leq y_i^\xi - x_j^\xi \;\; \forall \xi = 1, \ldots, k \\ \Leftrightarrow\; & a_{ij} \leq \bigwedge_{\xi=1}^{k} \left(y_i^\xi - x_j^\xi\right) = w_{ij} . \end{aligned} \tag{35}$$

This shows that

$$A \leq W \quad (\text{i.e., } a_{ij} \leq w_{ij} \;\forall i, j) \tag{36}$$

Equations 33 and 36 provide the following optimality criterion for W:
Whenever there exists a perfect recall memory A (i.e., $A \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi$ for $\xi = 1, \ldots, k$), then

$$A \leq W \quad \text{and} \quad W \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi \;\; \forall \xi = 1, \ldots, k. \tag{37}$$

The next obvious question, of course, concerns the existence of a perfect recall memory. Specifically, for what vector pairs $(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^k, \mathbf{y}^k)$ will W provide perfect recall? Once this question has been answered, the next logical question is to inquire as to the amount of noise W can tolerate for perfect recall; i.e., if $\tilde{\mathbf{x}}^\xi$ denotes a distorted version of $\mathbf{x}^\xi$, what are the conditions or bounds on $\tilde{\mathbf{x}}^\xi$ to insure that $W \boxtimes \tilde{\mathbf{x}}^\xi = \mathbf{y}^\xi$? We provide some answers to these questions. However, due to space limitations, we are not able to provide proofs here. The mathematical verification of these answers will be given elsewhere and are also available from the authors [21].

The following theorem answers the existence question of perfect recall memories for sets of pattern pairs.

Theorem 1. *$W \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi \;\; \forall \xi = 1, \ldots, k$ if and only if for each row index $i = 1, \ldots, m$ there exist column indices $j_\xi^i \in \{1, \ldots, n\}$, $(\xi \in \{1, \ldots, k\})$ such that*

$$w_{ij_\xi^i} = y_i^\xi - x_{j_\xi^i}^\xi \;\; \forall \xi = 1, \ldots, k . \tag{38}$$

We use the notation j_ξ^i to denote the fact that the column index j depends on the particular indices ξ and i.

The following is an easy consequence of this theorem.

Corollary 1. *$W \boxtimes \mathbf{x}^\xi = \mathbf{y}^\xi \;\; \forall \xi = 1, \ldots, k$ if and only if for each $\xi = 1, \ldots, k$ there exists subsets J_ξ of $\{1, \ldots, n\}$ having property $P(\xi)$, where $P(\xi)$ is the property: for each row index $i \in \{1, \ldots, m\}$ there exist column indices $j_\xi^i \in J_\xi$ which satisfy*

$$w_{ij_\xi^i} = y_i^\xi - x_{j_\xi^i}^\xi . \tag{39}$$

Note that the matrix W given in Example 1 satisfies the conditions of Theorem 1. For example, for $\xi = 2$ we have,

$$\begin{aligned} m_{11} &= -1 = -1 - 0 = y_1^2 - x_1^2, \\ m_{22} &= 1 = -1 - (-2) = y_2^2 - x_2^2, \\ m_{31} &= 0 = 0 - 0 = y_3^2 - x_1^2 . \end{aligned} \tag{40}$$

The following two theorems provide bounds for the amount of distortion of the exemplar patterns $\mathbf{x}^\xi$.

Theorem 2. *For a given $\gamma \in \{1, \ldots, k\}$ let J_γ be a subset of $\{1, \ldots, n\}$ with property $P(\gamma)$. If $\tilde{\mathbf{x}}^\gamma$ is a distorted version of the pattern $\mathbf{x}^\gamma$ with the property*

$$\begin{gathered} \tilde{x}_j^\gamma = x_j^\gamma \;\; \forall j \in J_\gamma \\ \text{and} \\ \tilde{x}_j^\gamma \leq x_j^\gamma \vee \bigwedge_{i=1}^{n} \left(\bigvee_{\xi \neq \gamma} \left[y_i^\gamma - y_i^\xi + x_j^\xi\right]\right) \\ \forall j \notin J_\gamma , \end{gathered} \tag{41}$$

then $W \boxtimes \tilde{\mathbf{x}}^\gamma = \mathbf{y}^\gamma$.

Theorem 3. *For $\gamma = 1, \ldots, k$ let $\tilde{\mathbf{x}}^\gamma$ denote the distorted version of the pattern $\mathbf{x}^\gamma$. Then $W \boxtimes \tilde{\mathbf{x}}^\gamma = \mathbf{y}^\gamma$ if and only if the following hold*

$$\tilde{x}_j^\gamma \leq x_j^\gamma \vee \bigwedge_{i=1}^{n} \left(\bigvee_{\xi \neq \gamma} \left[y_i^\gamma - y_i^\xi + x_j^\xi \right] \right) \quad \forall j = 1, \ldots, n$$

and if for each row index $i \in \{1, \ldots, m\}$ there exists a column index $j_i \in \{1, \ldots, n\}$ such that

$$\tilde{x}_{j_i}^\gamma = x_{j_i}^\gamma \vee \left(\bigvee_{\xi \neq \gamma} \left[y_i^\gamma - y_i^\xi + x_{j_i}^\xi \right] \right) .$$

A consequence of the proof of Theorem 1 [21] is that for given exemplar patterns $\mathbf{x}^\gamma$ the following equivalence holds:

$$\begin{aligned} & W \boxtimes \tilde{\mathbf{x}}^\gamma \leq \mathbf{y}^\gamma \\ \Leftrightarrow \quad & \tilde{x}_j^\gamma \leq x_j^\gamma \vee \bigwedge_{i=1}^{n} \left(\bigvee_{\xi \neq \gamma} \left[y_i^\gamma - y_i^\xi + x_j^\xi \right] \right) \qquad (44) \\ & \forall j = 1, \ldots, n . \end{aligned}$$

As a final note, we need to point out that the above results are concerned with morphological *perfect recall* memories. Similar to associative memories in the linear domain that may not have perfect recall capabilities (e.g., Hopfield nets) to distorted input, etc, recall in morphological associative memories can also be improved through the use of iterative steps and the introduction of appropriate threshold functions [21].

5. Single Layer Morphological Perceptrons

Let $\mathbb{Z}$ denote the set of integers and $\mathbb{Z}_{\pm\infty} = \mathbb{Z} \cup \{\infty, -\infty\}$. In analogy to a single layer linear perceptron, a single layer morphological perceptron is used to classify a pattern $\mathbf{p} = (p_1, p_2, \ldots, p_n) \in \mathbb{Z}^n$ (i.e., $p_i \in \mathbb{Z}$ for $i = 1, \ldots, n$) into one of two classes. A single layer morphological perceptron (SLMP) consists of a set of weights

$$W = \{w_1, w_2, \ldots, w_n\} \in \mathbb{Z}_{\pm\infty}^n \qquad (45)$$

and a hard limiter function $f : \mathbb{Z}_{\pm\infty} \to \{0, 1\}$ given by

$$f(x) = \begin{cases} 1 & if \ x > 0 \\ 0 & else . \end{cases} \qquad (46)$$

This formulation differs from the linear case where $n + 1$ weights are required.

In order to classify $\mathbf{p} \in \mathbb{Z}^n$, the perceptron first calculates the maximum of sums $g(\mathbf{p}) = \bigvee_{i=1}^{n} [p_i + w_i]$ and then applies the limiter function. If $f \circ g(\mathbf{p}) = 0$, the perceptron assigns $\mathbf{p}$ to class C_0. The pattern is assigned to class C_1 if $f \circ g(\mathbf{p}) = 1$.

The graph of $0 = g(\mathbf{x}) = \bigvee_{i=1}^{n} [x_i + w_i]$ is an $(n-1)$-dimensional figure that divides $\mathbb{Z}_{\pm\infty}^n$ into two regions. This area is called the perceptron's decision surface. In the 2–dimensional case, this decision surface corresponds to an infinite step function as shown in Figure 3. Note that a single layer linear perceptron cannot separate the two classes shown.

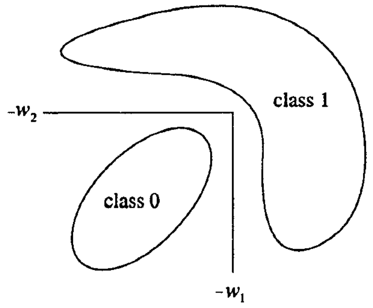

Fig. 3. Decision boundary for a single layer morphological perceptron.

Before the perceptron can act as a classifier, the values of its weights must be determined. If the decision surface is known a priori, the weights can be assigned analytically. Otherwise, the weights are determined in a learning stage, in which the perceptron determines its own decision surface through a learning algorithm. Unlike learning algorithms for conventional single layer perceptrons, the following learning algorithm for a single layer morphological perceptron has only a finite number of steps. Hence, the issue of convergence does not pose a problem.

Let $\{(\mathbf{p}^\xi, c_\xi)\}_{\xi=1}^{k}$ be a training set, where $\mathbf{p}^\xi$, $\xi = 1, \ldots, k$ are patterns in $\mathbb{Z}^n$, and $c_\xi \in \{0, 1\}$ are the class numbers associated with the patterns $\mathbf{p}^\xi$. The learning algorithm below finishes with the set of weights $w_i(k)$, $i = 1, \ldots, n$.

STEP 1. Set $w_i(0) = \infty$ for all $i = 1, \ldots, n$.
STEP 2. Adjust the weights k times according to the following rule: For $\xi = 1, \ldots, k$, compute

$$w_i(\xi) = \begin{cases} w_i(\xi - 1) & if \ f\left(p_i^\xi + w_i(\xi - 1)\right) = c_\xi \\ -p_i^\xi & if \ p_i^\xi + w_i(\xi - 1) > c_\xi \end{cases} \quad \forall i = 1, \ldots, n . \qquad (47)$$

If $f\left(\bigvee_{i=1}^{n} p_i^\xi + w_i(k) \right) < c_\xi$ for any $\xi = 1, \ldots, k$, then a single layer morphological perceptron is not able to assign the patterns p^ξ to the classes $c_\xi \in \{0, 1\}$ for all $\xi = 1, \ldots, k$.

Clearly, this concept of a single layer morphological perceptron presented above has the following advantages:

- Simple and fast classification mode due to the morphological operations used;
- finite learning mode; no convergence problems;
- numerical stability due to the fact that all basic operations are performed in $\mathbb{Z}_{\pm\infty}$.

However, this formulation of an SLMP only produces a very limited class of decision surfaces. In order to avoid these difficulties we introduced the concept of a generalized SLMP in which the computation of $g(\mathrm{p}) = \bigvee_{i=1}^{n} p_i + w_i$ is replaced by $g(\mathrm{p}) = a_0 \cdot \bigvee_{i=1}^{n} a_i(p_i + w_i)$ where $a_i \in \{-1, 1\}$ for $i = 0, 1, \ldots, n$. We established a learning rule for this perceptron and have shown that a three layer version of this net is able to create arbitrary decision surfaces in $\mathbb{Z}^n_{\pm\infty}$ [21].

REFERENCES

1. R.C. Backhouse and B. Carré. Regular algebra applied to path-finding problems. *J. Inst. Math. Appl.*, 15:161–186, 1975.
2. C. Benzaken. Structures algébra des cheminements. In G. Biorci, editor, *Network and Switching Theory*, pages 40–57. Academic Press, 1968.
3. B. Carré. An algebra for network routing problems. *J. Inst. Math. Appl.*, 7:273–294, 1971.
4. R. Cuninghame-Green. Process synchronisation in steelworks—a problem of feasibility. In Banbury and Maitland, editors, *Procceedings of the 2nd International Conference on Operations Research*, pages 323–328. English University Press, 1960.
5. R. Cuninghame-Green. Describing industrial processes with interference and approximating their steady-state behaviour. *Oper. Research Quart.*, 13:95–100, 1962.
6. R. Cuninghame-Green. *Minimax Algebra: Lecture Notes in Economics and Mathematical Systems 166*. Springer-Verlag, New York, 1979.
7. J.L. Davidson. *Lattice Structures in the Image Algebra and Applications to Image Processing*. PhD thesis, University of Florida, Gainesville, FL, 1989.
8. J.L. Davidson. Simulated annealing and morphological neural networks. In *Image Algebra and Morphological Image Processing III*, volume 1769 of *Proceedings of SPIE*, pages 119–127, San Diego, CA, July 1992.
9. J.L. Davidson and F. Hummer. Morphology neural networks: An introduction with applications. *IEEE Systems Signal Processing*, 12(2):177–210, 1993.
10. J.L. Davidson and G.X. Ritter. A theory of morphological neural networks. In *Digital Optical Computing II*, volume 1215 of *Proceedings of SPIE*, pages 378–388, July 1990.
11. J.L. Davidson and R. Srivastava. Fuzzy image algebra neural network for template identification. In *Second Annual Midwest Electro-Technology Conference*, pages 68–71, Ames, IA, April 1993. IEEE Central Iowa Section.
12. J.L. Davidson and A. Talukder. Template identification using simulated annealing in morphology neural networks. In *Second Annual Midwest Electro-Technology Conference*, pages 64–67, Ames, IA, April 1993. IEEE Central Iowa Section.
13. B. Giffler. Mathematical solution of production planning and scheduling problems. Tech. rep., IBM ASDD, 1960.
14. Z. Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, 1978.
15. V. Peteanu. An algebra of the optimal path in networks. *Mathematica*, 9:335–342, 1967.
16. G.X. Ritter. Recent developments in image algebra. In P. Hawkes, editor, *Advances in Electronics and Electron Physics*, volume 80, pages 243–308. Academic Press, New York, NY, 1991.
17. G.X. Ritter. Image algebra with applications. Unpublished manuscript, available via anonymous ftp from `ftp://ftp.cis.ufl.edu/pub/src/ia/documents`, 1994.
18. G.X. Ritter and J. L. Davidson. Recursion and feedback in image algebra. In *SPIE's 19th AIPR Workshop on Image Understanding in the 90's*, Proceedings of SPIE, McLean, Va., October 1990.
19. G.X. Ritter and J.L. Davidson. The image algebra and lattice theory. Technical Report TR 87–09, University of Florida CIS Department, 1987.
20. G.X. Ritter, D. Li, and J.N. Wilson. Image algebra and its relationship to neural networks. In *Technical Symposium Southeast on Optics, Electro-Optics, and Sensors*, Proceedings of SPIE, Orlando, FL, March 1989.
21. G.X. Ritter and P. Sussner. Morphological associative memories and perceptrons. Technical Report TR 96–02, CCVR, University of Florida, 1996.
22. G.X. Ritter and J.N. Wilson. *Handbook of Computer Vision Algorithms in Image Algebra*. CRC Press, Boca Raton, 1996.
23. G.X. Ritter, J.N. Wilson, and J.L. Davidson. Image algebra: An overview. *Computer Vision, Graphics, and Image Processing*, 49(3):297–331, March 1990.
24. J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
25. A. Shimbel. Structure in communication nets. In *Proceedings of the Symposium on Information Networks*, pages 119–203. Polytechnic Institute of Brooklyn, 1954.
26. C.P. Suarez-Araujo. Novel neural network models for homothetic invariances. *Journal of Mathematical Imaging and Vision*, 6(4):Preprint, 1996.

27. C.P. Suarez-Araujo and G.X. Ritter. Morphological neural networks and image algebra in artificial perception systems. In *Image Algebra and Morphological Image Processing III*, volume 1769 of *Proceedings of SPIE*, pages 128–142, San Diego, CA, July 1992.

28. S.S. Wilson. Morphological networks. In *Visual Communication and Image Processing IV*, Proceedings of SPIE, Philadelphia, PA, November 1989.