Morphological Perceptrons: Geometry and Training Algorithms

Vasileios Charisopoulos(✉) and Petros Maragos

School of ECE, National Technical University of Athens, 15773 Athens, Greece
vharisop@gmail.com, maragos@cs.ntua.gr

Abstract. Neural networks have traditionally relied on mostly linear models, such as the multiply-accumulate architecture of a linear perceptron that remains the dominant paradigm of neuronal computation. However, from a biological standpoint, neuron activity may as well involve inherently nonlinear and competitive operations. Mathematical morphology and minimax algebra provide the necessary background in the study of neural networks made up from these kinds of nonlinear units. This paper deals with such a model, called the morphological perceptron. We study some of its geometrical properties and introduce a training algorithm for binary classification. We point out the relationship between morphological classifiers and the recent field of tropical geometry, which enables us to obtain a precise bound on the number of linear regions of the maxout unit, a popular choice for deep neural networks introduced recently. Finally, we present some relevant numerical results.

Keywords: Mathematical morphology · Neural networks · Machine learning · Tropical geometry · Optimization

1 Introduction

In traditional literature on pattern recognition and machine learning, the so-called perceptron, introduced by Rosenblatt [21], has been the dominant model of neuronal computation. A *neuron* is a computational unit whose activation is a "multiply-accumulate" product of the input and a set of associated *synaptic weights*, optionally fed through a non-linearity. This model has been challenged in terms of both biological and mathematical plausibility by the morphological paradigm, widely used in computer vision and related disciplines. This has lately attracted a stronger interest from researchers in computational intelligence motivating further theoretical and practical advances in morphological neural networks, despite the fact that learning methods based on lattice algebra and mathematical morphology can be traced back to at least as far as the 90s (e.g. [6,19]).

In this paper, we re-visit the model of the *morphological perceptron* [24] in Sect. 3 and relate it with the framework of $(\max, +)$ and $(\min, +)$ algebras. In Sect. 3.1, we investigate its potential as a classifier, providing some fundamental geometric insight. We present a training algorithm for binary classification

J. Angulo et al. (Eds.): ISMM 2017, LNCS 10225, pp. 3–15, 2017.
DOI: 10.1007/978-3-319-57240-6_1

that uses the Convex-Concave Procedure and a more robust variant utilizing a simple form of outlier ablation. We also consider more general models such as maxout activations [11], relating the number of linear regions of a maxout unit with the *Newton Polytope* of its activation function, in Sect. 4. Finally, in Sect. 5, we present some experimental results pertinent to the efficiency of our proposed algorithm and provide some insight on the use of morphological layers in multilayer architectures.

We briefly describe the notation that we use. Denoting by $\mathbb{R}$ the line of real numbers, $(-\infty, \infty)$, let $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$ and $\mathbb{R}_{\min} = \mathbb{R} \cup \{\infty\}$. We use lowercase symbols for scalars (like x), lowercase symbols in boldface for vectors (like $\boldsymbol{w}$) and uppercase symbols in boldface for matrices (like $\boldsymbol{A}$). Vectors are assumed to be column vectors, unless explicitly stated otherwise.

We will focus on the $(\max, +)$ *semiring*, which is the semiring with underlying set $\mathbb{R}_{\max}$, using max as its binary "addition" and $+$ as its binary "multiplication". We may also refer to the $(\min, +)$ semiring which has an analogous definition, while the two semirings are actually isomorphic by the trivial mapping $\phi(x) = -x$. Both fall under the category of *idempotent* semirings [10], and are considered examples of so-called tropical semirings.[1]

Finally, we will use the symbol $\boxplus$ to refer to matrix and vector "multiplication" in $(\max, +)$ algebra and $\boxplus'$ for its dual in $(\min, +)$ algebra, following the convention established in [15]. Formally, we can define matrix multiplication as:

$$(\boldsymbol{A} \boxplus \boldsymbol{B})_{ij} = \bigvee_{q=1}^{k} A_{iq} + B_{qj} \qquad (\boldsymbol{A} \boxplus' \boldsymbol{B})_{ij} = \bigwedge_{q=1}^{k} A_{iq} + B_{qj} \tag{1}$$

for matrices of compatible dimensions.

2 Related Work

In [20], the authors argued about the biological plausibility of nonlinear responses, such as those introduced in Sect. 3. They proposed neurons computing max-sums and min-sums in an effort to mimic the response of a dendrite in a biological system, and showed that networks built from such neurons can approximate any compact region in Euclidean space within any desired degree of accuracy. They also presented a constructive algorithm for binary classification. Sussner and Esmi [24] introduced an algorithm based on competitive learning, combining morphological neurons to enclose training patterns in bounding boxes, achieving low response times and independence from the order by which training patterns are presented to the training procedure.

[1] The term "tropical" was playfully introduced by French mathematicians in honor of the Brazilian theoretical computer scientist, Imre Simon. Another example of a tropical semiring is the $(\max, \times)$ semiring, also referred to as the subtropical semiring.

Yang and Maragos [29] introduced the class of min-max classifiers, boolean-valued functions appearing as thresholded minima of maximum terms or maxima of minimum terms:

$$f_{\text{max-min}}(x_1, x_2, \dots x_d) = \bigwedge_j \bigvee_{i \in I_j} l_i, \quad l_i \in \{x_i, 1 - x_i\} \tag{2}$$

and vice-versa for $f_{\text{min-max}}$. In the above, I_j is the set of indices corresponding to term j. These classifiers produce decision regions similar to those formed by a $(\max, +)$ or $(\min, +)$ perceptron.

Barrera et al. [3] tried to tackle the problem of statistically optimal design for set operators on binary images, consisting of morphological operators on sets. They introduced an *interval splitting* procedure for learning boolean concepts and applied it to binary image analysis, such as edge detection or texture recognition.

With the exception of [29], the above introduce constructive training algorithms which may produce complex decision regions, as they fit models precisely to the training set. They may create superfluous decision areas to include outliers that might be disregarded when using gradient based training methods, a fact that motivates the work in Sect. 3.2.

In a recent technical report, Gärtner and Jaggi [8] proposed the concept of a *tropical support vector machine*. Its response and j-th decision region are given by:

$$y(\boldsymbol{x}) = \bigwedge_{i=1}^{n} w_i + x_i \ , \quad \mathcal{R}^j(\boldsymbol{x}) = \{\boldsymbol{x} : w_j + x_j \leq w_i + x_i, \forall i\} \tag{3}$$

instead of a "classical" decision region (e.g. defined by some discriminant function).

Cuninghame-Green's work on minimax algebra [5] provides much of the matrix-vector framework for the finite-dimensional morphological paradigm. A fundamental result behind Sussner and Valle's article [25] on morphological analogues of classical associative memories such as the Hopfield network, states that the "closest" under-approximation of a target vector $\boldsymbol{b}$ by a max-product in the form $\boldsymbol{A} \boxplus \boldsymbol{x}$ can be found by the so-called *principal solution* of a max-linear equation.

Theorem 1. [5] *If* $\boldsymbol{A} \in \mathbb{R}_{\max}^{m \times n}, \boldsymbol{b} \in \mathbb{R}_{\max}^{m}$, *then*

$$\overline{\boldsymbol{x}} = \boldsymbol{A}^\sharp \boxplus' \boldsymbol{b} \qquad (\boldsymbol{A}^\sharp \triangleq -\boldsymbol{A}^T) \tag{4}$$

is the greatest solution to $\boldsymbol{A} \boxplus \boldsymbol{x} \leq \boldsymbol{b}$, *and furthermore* $\boldsymbol{A} \boxplus \boldsymbol{x} = \boldsymbol{b}$ *has a solution if and only if* $\overline{\boldsymbol{x}}$ *is a solution.*[2]

[2] The matrix $-\boldsymbol{A}^T$, often denoted by $\boldsymbol{A}^\sharp$ in the tropical geometry community, is sometimes called the *Cuninghame-Green inverse* of $\boldsymbol{A}$.

3 The Morphological Perceptron

Classical literature defines the perceptron as a computational unit with a linear activation possibly fed into a non-linearity. Its output is the result of the application of an activation function, that is usually nonlinear, to its activation $\phi(\boldsymbol{x})$. Popular examples are the logistic sigmoid function or the rectifier linear unit, which has grown in popularity among deep learning practitioners [17]. For the morphological neuron, in [20], its response to an input $\boldsymbol{x} \in \mathbb{R}^n$ is given by

$$\tau(\boldsymbol{x}) = p \cdot \bigvee_{i=1}^{n} r_i(x_i + w_i) \ , \qquad \tau'(\boldsymbol{x}) = p \cdot \bigwedge_{i=1}^{n} r_i(x_i + m_i) \tag{5}$$

for the cases of the $(\max, +)$ and $(\min, +)$ semirings respectively. Parameters r_i and p take values in $\{+1, -1\}$ depending on whether the synapses and the output are excitatory or inhibitory. We adopt a much simpler version:

Definition 1. (Morphological Perceptron). *Given an input vector $\boldsymbol{x} \in \mathbb{R}^n_{\max}$, the morphological perceptron associated with weight vector $\boldsymbol{w} \in \mathbb{R}^n_{\max}$ and activation bias $w_0 \in \mathbb{R}_{\max}$ computes the activation*

$$\tau(\boldsymbol{x}) = w_0 \vee (w_1 + x_1) \vee \cdots \vee (w_n + x_n) = w_0 \vee \left(\bigvee_{i=1}^{n} w_i + x_i \right) \tag{6}$$

We may define a "dual" model on the $(\min, +)$ semiring, as the perceptron with parameters $\boldsymbol{m} \in \mathbb{R}^n_{\min}, m_0 \in \mathbb{R}_{\min}$ that computes the activation

$$\tau'(\boldsymbol{x}) = m_0 \wedge (m_1 + x_1) \wedge \cdots \wedge (m_n + x_n) = m_0 \wedge \left(\bigwedge_{i=1}^{n} m_i + x_i \right) \tag{7}$$

The models defined by (6, 7) may also be referred to as $(\max, +)$ and $(\min, +)$ perceptron, respectively. They can be treated as instances of morphological filters [14,22], as they define a (grayscale) dilation and erosion over a finite window, computed at a certain point in space or time. Note that $\tau(\boldsymbol{x})$ is a nonlinear, convex (as piecewise maximum of affine functions) function of $\boldsymbol{x}, \boldsymbol{w}$ that is continuous everywhere, but not differentiable everywhere (points where multiple terms maximize $\tau(\boldsymbol{x})$ are singular).

3.1 Geometry of a $(\max, +)$ Perceptron for Binary Classification

Let us now put the morphological perceptron into the context of binary classification. We will first try to investigate the perceptron's geometrical properties drawing some background from tropical geometry.

Let $\boldsymbol{X} \in \mathbb{R}^{k \times n}_{\max}$ be a matrix containing the patterns to be classified as its rows, let $\boldsymbol{x}^{(k)}$ denote the k-th pattern (row) and let $\mathcal{C}_1, \mathcal{C}_0$ be the two classes of the relevant decision problem. Without loss of generality, we may choose

$y_k = 1$ if $\boldsymbol{x}^{(k)} \in \mathcal{C}_1$ and $y_k = -1$ if $\boldsymbol{x}^{(k)} \in \mathcal{C}_0$. Using the notation in (1), the $(\max, +)$ perceptron with parameter vector $\boldsymbol{w}$ computes the output

$$\tau(\boldsymbol{x}) = \boldsymbol{w}^T \boxplus \boldsymbol{x} \tag{8}$$

Note that the variant we study here has no activation bias ($w_0 = -\infty$). If we assign class labels to patterns based on the sign function, we have $\tau(\boldsymbol{x}) > 0 \Rightarrow \boldsymbol{x} \in \mathcal{C}_1$, $\tau(\boldsymbol{x}) < 0 \Rightarrow \boldsymbol{x} \in \mathcal{C}_0$. Therefore, the decision regions formed by that perceptron have the form

$$\mathcal{R}_1 := \{\boldsymbol{x} \in \mathbb{R}_{\max}^n : \boldsymbol{w}^T \boxplus \boldsymbol{x} \geq 0\},\ \mathcal{R}_0 := \{\boldsymbol{x} \in \mathbb{R}_{\max}^n : \boldsymbol{w}^T \boxplus \boldsymbol{x} \leq 0\} \tag{9}$$

As it turns out, these inequalities are collections of so called *affine tropical halfspaces* and define *tropical polyhedra* [9,13], which we will now introduce.

Definition 2 (Affine tropical halfspace). *Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}_{\max}^{n+1}$. An **affine tropical halfspace** is a subset of $\mathbb{R}_{\max}^n$ defined by*

$$\mathcal{T}(\boldsymbol{a}, \boldsymbol{b}) := \left\{ \boldsymbol{x} \in \mathbb{R}_{\max}^n : \left(\bigvee_{i=1}^{n} a_i + x_i \right) \vee a_{n+1} \geq \left(\bigvee_{i=1}^{n} b_i + x_i \right) \vee b_{n+1} \right\} \tag{10}$$

We can further assume that $\min(a_i, b_i) = -\infty \quad \forall i \in \{1, 2, \ldots, n+1\}$, as per [9, Lemma 1].

A *tropical polyhedron* is the intersection of finitely many tropical halfspaces (and comes in signed and unsigned variants, as in [1]). In our context, we will deal with tropical polyhedra like the following: assume $\boldsymbol{A} \in \mathbb{R}_{\max}^{m \times n}$, $\boldsymbol{B} \in \mathbb{R}_{\max}^{k \times n}$, $\boldsymbol{c} \in \mathbb{R}_{\max}^m$ and $\boldsymbol{d} \in \mathbb{R}_{\max}^k$. The inequalities

$$\boldsymbol{A} \boxplus \boldsymbol{x} \geq \boldsymbol{c}\ , \quad \boldsymbol{B} \boxplus \boldsymbol{x} \leq \boldsymbol{d} \tag{11}$$

define a subset $\mathcal{P} \subseteq \mathbb{R}_{\max}^n$ that is a tropical polyhedron, which can be empty if some of the inequalities cannot be satisfied, leading us to our first remark.

Proposition 1 (Feasible Regions are Tropical Polyhedra). *Let $\boldsymbol{X} \in \mathbb{R}_{\max}^{k \times n}$ be a matrix containing input patterns of dimension n as its rows, partitioned into two distinct matrices $\boldsymbol{X}_{\text{pos}}$ and $\boldsymbol{X}_{\text{neg}}$, which contain all patterns of classes $\mathcal{C}_1, \mathcal{C}_0$ respectively. Let $\mathcal{T}$ be the tropical polyhedron defined by*

$$\mathcal{T}(\boldsymbol{X}_{\text{pos}}, \boldsymbol{X}_{\text{neg}}) = \{\boldsymbol{w} \in \mathbb{R}_{\max}^n : \boldsymbol{X}_{\text{pos}} \boxplus \boldsymbol{w} \geq 0,\ \boldsymbol{X}_{\text{neg}} \boxplus \boldsymbol{w} \leq 0\} \tag{12}$$

Patterns $\boldsymbol{X}_{\text{pos}}, \boldsymbol{X}_{\text{neg}}$ can be completely separated by a $(\max, +)$ perceptron if and only if $\mathcal{T}$ is nonempty.

Remark 1. In [9], it has been shown that the question of a tropical polyhedron being nonempty is polynomially equivalent to an associated mean payoff game having a winning initial state.

Using the notion of the *Cuninghame-Green inverse* from Theorem 1, we can restate the separability condition in Proposition 1. As we know that $\overline{\boldsymbol{w}} = \boldsymbol{X}_{\text{neg}}^{\sharp} \boxplus' \boldsymbol{0}$ is the greatest solution to $\boldsymbol{X}_{\text{neg}} \boxplus \boldsymbol{w} \leq \boldsymbol{0}$, that condition is equivalent to

$$\boldsymbol{X}_{\text{pos}} \boxplus (\boldsymbol{X}_{\text{neg}}^{\sharp} \boxplus' \boldsymbol{0}) \geq \boldsymbol{0} \tag{13}$$

3.2 A Training Algorithm Based on the Convex-Concave Procedure

In this section, we present a training algorithm that uses the Convex-Concave Procedure [30] in a manner similar to how traditional Support Vector Machines use convex optimization to determine the optimal weight assignment for a binary classification problem. It is possible to state an optimization problem with a convex cost function and constraints that consist of inequalities of difference-of-convex (DC) functions. Such optimization problems can be solved (at least approximately) by the Convex-Concave Procedure.

$$\begin{aligned} &\text{Minimize } J(\boldsymbol{X}, \boldsymbol{w}) = \sum_{k=1}^{K} \max(\xi_k, 0) \\ &\text{s. t. } \begin{cases} \bigvee_{i=1}^{n} w_i + x_i^{(k)} \leq \xi_k & \text{if } \boldsymbol{x}^{(k)} \in \mathcal{C}_0 \\ \bigvee_{i=1}^{n} w_i + x_i^{(k)} \geq -\xi_k & \text{if } \boldsymbol{x}^{(k)} \in \mathcal{C}_1 \end{cases} \end{aligned} \tag{14}$$

The slack variables ξ_k in the constraints are used to ensure that only misclassified patterns will contribute to J. In our implementation, we use [23, Algorithm 1.1], utilizing the authors' `DCCP` library that extends CvxPy [7], a modelling language for convex optimization in `Python`. An application on separable patterns generated from a Gaussian distribution can be seen in Fig. 1.

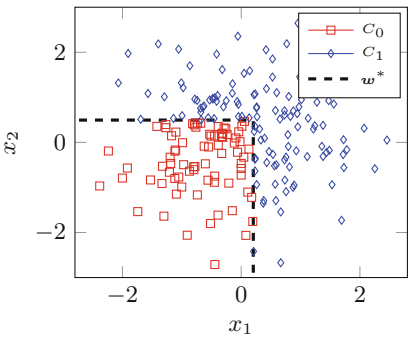

Fig. 1. Decision surface

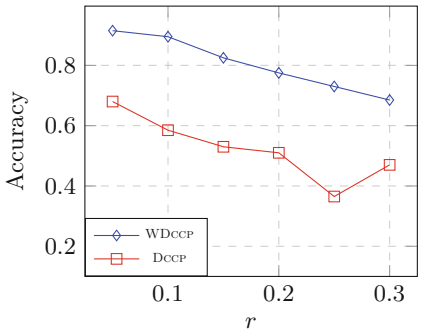

Fig. 2. Method accuracy

So far, we have not addressed the case where patterns are not separable or contain "abnormal" entries and outliers. Although many ways have been proposed to deal with the presence of outliers [28], the method we used to overcome this was to "penalize" patterns with greater chances of being outliers. We introduce a simple weighting scheme that assigns, to each pattern, a factor that is inversely proportional to its distance (measured by some ℓ_p-norm) from its class's centroid.

$$\boldsymbol{\mu}_i := \frac{1}{|\mathcal{C}_i|} \sum_{\boldsymbol{x}^{(k)} \in \mathcal{C}_i} \boldsymbol{x}^{(k)}, \;\; \lambda_k := \frac{1}{||\boldsymbol{x}^{(k)} - \boldsymbol{\mu}_i||_p} \tag{15}$$

$$\nu_k := \frac{\lambda_k}{\max_k \lambda_k} \tag{16}$$

Equation (16) above serves as a normalization step that scales all λ_k in the $(0, 1]$ range. We arrive at a reformulated optimization problem which can be stated as

$$\begin{aligned} &\text{Minimize } J(\boldsymbol{X}, \boldsymbol{w}, \boldsymbol{\nu}) = \sum_{k=1}^{K} \nu_k \cdot \max(\xi_k, 0) \\ &\text{s. t.} \begin{cases} \bigvee_{i=1}^{n} w_i + x_i^{(k)} \leq \xi_k & \text{if } \boldsymbol{x}^{(k)} \in \mathcal{C}_0 \\ \bigvee_{i=1}^{n} w_i + x_i^{(k)} \geq -\xi_k & \text{if } \boldsymbol{x}^{(k)} \in \mathcal{C}_1 \end{cases} \end{aligned} \tag{17}$$

To illustrate the practical benefits of this method (which we will refer to as WDCCP), we use both versions of the optimization problem on a set of randomly generated data which is initially separable but then a percentage r of its class labels is flipped. Comparative results for a series of percentages r are found in Fig. 2. The results for $r = 20\%$ can be seen in Fig. 3, with the dashed line representing the weights found by WDCCP. This weighting method can be extended to complex or heterogeneous data; for example, one could try and fit a set of patterns to a mixture of Gaussians or perform clustering to obtain the coefficients $\boldsymbol{\nu}$.

It is possible to generalize the morphological perceptron to combinations of dilations (max-terms) and erosions (min-terms). In [2], the authors introduce the *Dilation-Erosion Linear Perceptron*, which contains a convex combination of a dilation and an erosion, as:

$$M(\boldsymbol{x}) = \lambda\tau(\boldsymbol{x}) + (1 - \lambda)\tau'(\boldsymbol{x}), \quad \lambda \in [0, 1] \tag{18}$$

plus a linear term, employing gradient descent for training. The formulation in (17) can be used here too, as constraints in difference-of-convex programming can be (assuming f_i convex):

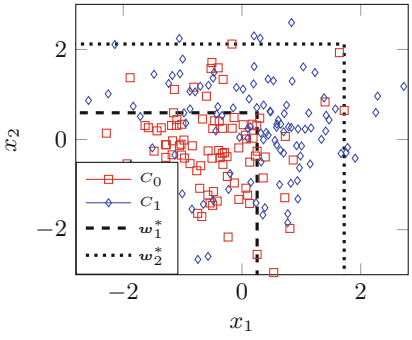

Fig. 3. Optimal weights found

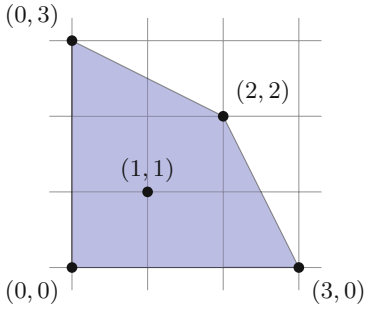

Fig. 4. Newt(p) of Eq. (23)

$$f_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \leq 0,\ g_i \text{ convex, or } f_i(\boldsymbol{x}) + g_i'(\boldsymbol{x}) \leq 0,\ g_i' \text{ concave} \tag{19}$$

This observation is exploited in the first experiment of Sect. 5.

4 Geometric Interpretation of Maxout Units

Maxout units were introduced by Goodfellow et al. [11]. A maxout unit is associated with a weight matrix $\boldsymbol{W} \in \mathbb{R}_{\max}^{k \times n}$ as well as an activation bias vector $\boldsymbol{b} \in \mathbb{R}_{\max}^{k}$. Given an input pattern $\boldsymbol{x} \in \mathbb{R}_{\max}^{n}$ and denoting by $\boldsymbol{W}_{j,:}$ the j-th row vector of $\boldsymbol{W}$, a maxout unit computes the following activation:

$$h(\boldsymbol{x}) = \bigvee_{j=1}^{k} \boldsymbol{W}_{j,:}\boldsymbol{x} + b_j = \bigvee_{j=1}^{k} \left[\left(\sum_{i=1}^{n} W_{ji} x_i \right) + b_j \right] \tag{20}$$

Essentially, a maxout unit generalizes the morphological perceptron using k terms (referred to as the unit's *rank*) that involve affine expressions. In tropical algebra, such expressions are called *tropical polynomials* [13] or *maxpolynomials* [4] when specifically referring to the $(\max, +)$ semiring. In [16], maxout units are investigated geometrically in an effort to obtain bounds for the number of linear regions of a deep neural network with maxout layers:

Proposition 2 ([16], Proposition 7). *The maximal number of linear regions of a single layer maxout network with n inputs and m outputs of rank k is lower bounded by $k^{\min(n,m)}$ and upper bounded by* $\min\left\{ \sum_{j=0}^{n} \binom{k^2 m}{j}, k^m \right\}$.

This result readily applies to layers consisting of $(\max, +)$ perceptrons, as a $(\max, +)$ perceptron has rank $k = n$.

For a maxout unit of rank k, the authors argued that the number of its linear regions is exactly k if every term is maximal at some point. We provide an exact result using tools from tropical geometry; namely, the *Newton Polytope* of a maxpolynomial. For definitions and fundamental results on polytopes the reader is referred to [31]; we kick off our investigation omitting the presence of the bias term b_j as seen in (20).

Definition 3 (Newton Polytope). *Let $p : \mathbb{R}_{\max}^{n} \to \mathbb{R}_{\max}$ be a maxpolynomial with k terms, given by*

$$p(\boldsymbol{x}) = \max_{i \in 1,2,\ldots,k} \{c_{i1}x_1 + c_{i2}x_2 + \cdots + c_{in}x_n\} = \bigvee_{i=1}^{k} \boldsymbol{c}_i^T \boldsymbol{x} \tag{21}$$

The Newton Polytope of p is the convex hull of the coefficient vectors $\boldsymbol{c}_i$:

$$\text{Newt}(p) = \text{conv}\{\boldsymbol{c}_i : i \in 1, \ldots, k\} = \text{conv}\{(c_{i1}, c_{i2}, \ldots c_{in}) : i \in 1, \ldots, k\} \tag{22}$$

For an illustrative example, see Fig. 4. The maxpolynomial in question is

$$p(\boldsymbol{x}) = 0 \vee (x + y) \vee 3x \vee (2x + 2y) \vee 3y \tag{23}$$

and its terms can be matched to the coefficient vectors $(0, 0), (1, 1), (3, 0), (2, 2)$ and $(0, 3)$ respectively. The Newton Polytope's vertices give us information about the number of linear regions of the associated maxpolynomial:

Proposition 3. *Let $p(\boldsymbol{x})$ be a maxout unit with activation given by (21). The number of p's linear regions is equal to the number of vertices of its Newton Polytope,* $\mathrm{Newt}(p)$.

Proof. A proof can be given using the fundamental theorem of Linear Programming [26, Theorem 3.4]. Consider the linear program:

$$\begin{aligned} &\text{Maximize } \boldsymbol{x}^T \boldsymbol{c} \\ &\text{s.t. } \boldsymbol{c} \in \mathrm{Newt}(p) \end{aligned} \tag{24}$$

Note that, for our purposes, $\boldsymbol{c}$ is the variable to be optimized. Letting $\boldsymbol{c}$ run over assignments of coefficient vectors, we know that for every $\boldsymbol{x}$, Problem (24) is a linear program for which the maximum is attained at one of the vertices of $\mathrm{Newt}(p)$. Therefore, points $\boldsymbol{c}_i \in \mathrm{int}(\mathrm{Newt}(p))$ map to coefficient vectors of non-maximal terms of p. □

By Proposition 3, we conclude that the term $x + y$ can be omitted from $p(\boldsymbol{x})$ in (23) without altering it as a function of $\boldsymbol{x}$. Proposition 3 can be extended to maxpolynomials with constant terms, such as maxout units with bias terms b_j. Let the extended Newton Polytope be

$$p(\boldsymbol{x}) = \bigvee_{j=1}^{k} b_j + \boldsymbol{c}_j^T \boldsymbol{x} \Rightarrow \mathrm{Newt}(p) = \mathrm{conv}\left\{(b_j, \boldsymbol{c}_j) : j \in 1, \ldots, k\right\} \tag{25}$$

Let $\boldsymbol{c}' = (b, \boldsymbol{c})$ and $\boldsymbol{x}' = (1, \boldsymbol{x})$. Note that the relevant linear program is now

$$\begin{aligned} &\text{Maximize } (\boldsymbol{x}')^T \boldsymbol{c}' \\ &\text{s.t. } \boldsymbol{c}' \in \mathrm{Newt}(p) \end{aligned} \tag{26}$$

The optimal solutions of this program lie in the *upper hull* of $\mathrm{Newt}(p)$, $\mathrm{Newt}^{\max}(p)$, with respect to b. For a convex polytope P, its upper hull is

$$P^{\max} := \{(\lambda, \boldsymbol{x}) \in P : (t, \boldsymbol{x}) \in P \Rightarrow t \leq \lambda\} \tag{27}$$

Therefore, the number of linear regions of a maxout unit given by (20) is equal to the number of vertices on the upper hull of its Newton Polytope. Those results are easily extended for the following models:

Proposition 4. *Let $h_1, \dots h_m$ be a collection of maxpolynomials. Let*

$$g_\vee(\boldsymbol{x}) = \bigvee_{i=1}^{m} h_i(\boldsymbol{x}), \qquad g_+(\boldsymbol{x}) = \sum_{i=1}^{m} h_i(\boldsymbol{x}) \tag{28}$$

The Newton Polytopes of the functions defined above are

$$\operatorname{Newt}(g_\vee) = \operatorname{conv}(\operatorname{Newt}(h_1), \dots \operatorname{Newt}(h_m)) \tag{29}$$

$$\operatorname{Newt}(g_+) = \operatorname{Newt}(h_1) \oplus \operatorname{Newt}(h_2) \cdots \oplus \operatorname{Newt}(h_m) \tag{30}$$

where $\oplus$ denotes the Minkowski sum of the Newton Polytopes.

5 Experiments

In this section, we present results from a few numerical experiments conducted to examine the efficiency of our proposed algorithm and the behavior of morphological units as parts of a multilayer neural network.

5.1 Evaluation of the WDCCP Method

Our first experiment uses a dilation-erosion or *max-min* morphological perceptron, whose response is given by

$$y(\boldsymbol{x}) = \lambda \left(\bigvee_{i=1}^{n} w_i + x_i \right) + (1 - \lambda) \left(\bigwedge_{i=1}^{n} m_i + x_i \right) \tag{31}$$

We set $\lambda = 0.5$ and trained it using both stochastic gradient descent with MSE cost and learning rate η (SGD) as well as the WDCCP method on Ripley's Synthetic Dataset [18] and the Wisconsin Breast Cancer Dataset [27]. Both are 2-class, non-separable datasets. For simplicity, we fixed the number of epochs for the gradient method at 100 and set $\tau_{\max} = 0.01$ and stopping criterion $\epsilon \leq 10^{-3}$ for the WDCCP method. We repeated each experiment 50 times to obtain mean and standard deviation for its classification accuracy, shown in Table 1. On all cases, the WDCCP method required less than 10 iterations to converge and exhibited far better results than gradient descent. The negligible standard deviation of its accuracy hints towards robustness in comparison to other methods.

5.2 Layers Using Morphological Perceptrons

We experimented on the MNIST dataset of handwritten digits [12] to investigate how morphological units behave when incorporated in layers of neural networks. After some unsuccessful attempts using a single-layer network, we settled on the following architecture: a layer of n_1 linear units followed by a $(\max, +)$ output layer of 10 units with softmax activations. The case for $n_1 = 64$ is illuminating,

Table 1. `Ripleys`/`WDBC` test set results

η	`Ripleys`		`WDBC`	
	SGD	WDCCP	SGD	WDCCP
0.01	0.838 ± 0.011	**0.902** ± 0.001	0.726 ± 0.002	**0.908** ± 0.001
0.02	0.739 ± 0.012		0.763 ± 0.006	
0.03	0.827 ± 0.008		0.726 ± 0.004	
0.04	0.834 ± 0.008		0.751 ± 0.007	
0.05	0.800 ± 0.009		0.783 ± 0.012	
0.06	0.785 ± 0.008		0.768 ± 0.01	
0.07	0.776 ± 0.009		0.729 ± 0.009	
0.08	0.769 ± 0.01		0.732 ± 0.01	
0.09	0.799 ± 0.009		0.730 ± 0.015	
0.1	0.749 ± 0.011		0.729 ± 0.009	

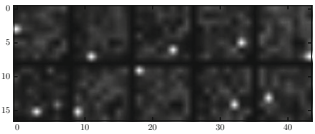

Fig. 5. Dilation layer

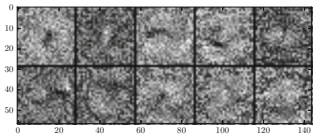

Fig. 6. Active filters

as we decided to plot the morphological filters as grayscale images shown in Fig. 5. Plotting the linear units resulted in noisy images except for those shown in Fig. 6, corresponding to maximal weights in the dilation layer. The dilation layer takes into account just one or two linear activation units per digit (pictured as bright dots), so we re-evaluated the accuracy after "deactivating" the rest of them, obtaining the same accuracy, as shown in Table 2.

Table 2. MNIST results

Layer n_1	Accuracy	Accuracy without "dead" units	# Active filters
24	84.29%	84.28%	17
32	84.84%	84.85%	15
48	84.63%	84.61%	18
64	92.1%	92.07%	10

6 Conclusions and Future Work

In this paper, we examined some properties and the behavior of morphological classifiers and introduced a training algorithm based on a well-studied optimization problem. We aim to further investigate the potential of both ours and other models, such as that proposed in [8]. A natural next step would be to examine their performance as parts of deeper architectures, possibly taking advantage of their tendency towards sparse activations to simplify the resulting networks.

The subtle connections with tropical geometry that we were able to identify make us believe that it could also aid others in the effort to study fundamental properties of deep, nonlinear architectures. We hope that the results of this paper will further motivate researchers active in those areas towards that end.

Acknowledgements. This work was partially supported by the European Union under the projects BabyRobot with grant H2020-687831 and I-SUPPORT with grant H2020-643666.

References

1. Allamigeon, X., Benchimol, P., Gaubert, S., Joswig, M.: Tropicalizing the simplex algorithm. SIAM J. Discret. Math. **29**(2), 751–795 (2015)
2. Araújo, R.D.A., Oliveira, A.L., Meira, S.R: A hybrid neuron with gradient-based learning for binary classification problems. In: Encontro Nacional de Inteligência Artificial-ENIA (2012)
3. Barrera, J., Dougherty, E.R., Tomita, N.S.: Automatic programming of binary morphological machines by design of statistically optimal operators in the context of computational learning theory. J. Electron. Imaging **6**(1), 54–67 (1997)
4. Butkovič, P.: Max-linear Systems: Theory and Algorithms. Springer Science & Business Media, Heidelberg (2010)
5. Cuninghame-Green, R.A.: Minimax Algebra. Lecture Notes in Economics and Mathematical Systems, vol. 166. Springer, Heidelberg (1979)
6. Davidson, J.L., Hummer, F.: Morphology neural networks: an introduction with applications. Circ. Syst. Sig. Process. **12**(2), 177–210 (1993)
7. Diamond, S., Boyd, S.: CVXPY: a Python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)
8. Gärtner, B., Jaggi, M.: Tropical support vector machines. Technical report ACS-TR-362502-01 (2008)
9. Gaubert, S., Katz, R.D.: Minimal half-spaces and external representation of tropical polyhedra. J. Algebraic Comb. **33**(3), 325–348 (2011)
10. Gondran, M., Minoux, M.: Graphs, Dioids and Semirings: New Models and Algorithms, vol. 41. Springer Science & Business Media, Heidelberg (2008)
11. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A.C., Bengio, Y.: Maxout networks. ICML **3**(28), 1319–1327 (2013)
12. LeCun, Y., Cortes, C., Burges, C.J.: The MNIST database of handwritten digits (1998)
13. Maclagan, D., Sturmfels, B.: Introduction to Tropical Geometry, vol. 161. American Mathematical Society, Providence (2015)
14. Maragos, P.: Morphological filtering for image enhancement and feature detection. In: Bovik, A.C. (ed.) The Image and Video Processing Handbook, 2nd edn, pp. 135–156. Elsevier Academic Press, Amsterdam (2005)
15. Maragos, P.: Dynamical systems on weighted lattices: general theory. arXiv preprint arXiv:1606.07347 (2016)
16. Montufar, G.F., Pascanu, R., Cho, K., Bengio, Y.: On the number of linear regions of deep neural networks. In: Advances in Neural Information Processing Systems, pp. 2924–2932 (2014)
17. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML 2010, pp. 807–814 (2010)
18. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (2007)
19. Ritter, G.X., Sussner, P.: An introduction to morphological neural networks. In: 1996 Proceedings of the 13th International Conference on Pattern Recognition, vol. 4, pp. 709–717. IEEE (1996)

20. Ritter, G.X., Urcid, G.: Lattice algebra approach to single-neuron computation. IEEE Trans. Neural Netw. **14**(2), 282–295 (2003)
21. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**(6), 386 (1958)
22. Serra, J.: Image Analysis and Mathematical Morphology, vol. 1. Academic Press, Cambridge (1982)
23. Shen, X., Diamond, S., Gu, Y., Boyd, S.: Disciplined convex-concave programming. arXiv preprint arXiv:1604.02639 (2016)
24. Sussner, P., Esmi, E.L.: Morphological perceptrons with competitive learning: lattice-theoretical framework and constructive learning algorithm. Inf. Sci. **181**(10), 1929–1950 (2011)
25. Sussner, P., Valle, M.E.: Gray-scale morphological associative memories. IEEE Trans. Neural Netw. **17**(3), 559–570 (2006)
26. Vanderbei, R.J., et al.: Linear Programming. Springer, Heidelberg (2015)
27. Wolberg, W.H., Mangasarian, O.L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proc. Nat. Acad. Sci. **87**(23), 9193–9196 (1990)
28. Xu, L., Crammer, K., Schuurmans, D.: Robust support vector machine training via convex outlier ablation. In: AAAI, vol. 6, pp. 536–542 (2006)
29. Yang, P.F., Maragos, P.: Min-max classifiers: learnability, design and application. Pattern Recogn. **28**(6), 879–899 (1995)
30. Yuille, A.L., Rangarajan, A.: The concave-convex procedure. Neural Comput. **15**(4), 915–936 (2003)
31. Ziegler, G.M.: Lectures on Polytopes, vol. 152. Springer Science & Business Media, Heidelberg (1995)