Please cite as: I. Kordonis and P. Maragos, "Revisiting Tropical Polynomial Division: Theory, Algorithms, and Application to Neural Networks", *IEEE Transactions on Neural Networks and Learning Systems*, 2025, DOI: https://doi.org/10.1109/TNNLS.2025.3570807 .

Revisiting Tropical Polynomial Division: Theory, Algorithms, and Application to Neural Networks

Ioannis Kordonis, Petros Maragos, *Life Fellow IEEE*

***Abstract*—Tropical geometry has recently found several applications in the analysis of neural networks with piecewise linear activation functions. This paper presents a new look at the problem of tropical polynomial division and its application to the simplification of neural networks. We analyze tropical polynomials with real coefficients, extending earlier ideas and methods developed for polynomials with integer coefficients. We first prove the existence of a unique quotient-remainder pair and characterize the quotient in terms of the convex bi-conjugate of a related function. Interestingly, the quotient of tropical polynomials with integer coefficients does not necessarily have integer coefficients. Furthermore, we develop a relationship of tropical polynomial division with the computation of the convex hull of unions of convex polyhedra and use it to derive an exact algorithm for tropical polynomial division. An approximate algorithm is also presented, based on an alternation between data partition and linear programming. We also develop special techniques to divide composite polynomials, described as sums or maxima of simpler ones. Finally, we provide numerical results to demonstrate the efficiency of the proposed algorithms, using the MNIST handwritten digits, SVHN, CIFAR-10, and CIFAR-100 datasets, along with an application example in Learning Model Predictive Control.**

***Index Terms*— Tropical geometry, Piecewise linear neural networks, Tropical polynomial division, Neural network compression**

I. Introduction

Tropical geometry is a relatively new research field combining elements and ideas from polyhedral and algebraic geometry [1]. The underlying algebraic structure is the max-plus semiring (also known as tropical semiring), where the usual addition is replaced by maximization and the standard multiplication by addition. Tropical polynomials (also known as max-polynomials) are the polynomials in the max-plus algebra and have a central role in tropical geometry. This work deals with the division of tropical polynomials and presents some applications in neural network compression.

A link between tropical geometry and machine learning was recently developed [2], [3]. An important application is the analysis of neural networks with piecewise linear activations (e.g., ReLU). In this front, [3], [4], [5], [6], [7], [8] use a tropical representation of neural networks to describe the complexity of a network structure, defined as the number of its linear regions, or describe their decision boundaries. [9] presents an algorithm to extract the linear regions of deep ReLU neural networks. Papers [10], [11], [12] deal with the problem of tropical polynomial division and its use in the simplification of neural networks. The analysis is, however, restricted to polynomials with integer coefficients. Articles [13], [14] use tropical representations of neural networks and employs polytope approximation tools to simplify them. Another class of networks represented in terms of tropical algebra is morphological neural networks [15], [16], [17], [18], [19], [20]. Other applications of tropical algebra and geometry include the tropical modeling of classical algorithms in probabilistic graphical models [21], [22] and piecewise linear regression [23], [24]. Neural networks employing the closely related log-sum-exp nonlinearity were presented in [25], [26].

Another problem very much related to the current work is the factorization of tropical polynomials. This problem was first studied in [27], [28], [29], [30], for the single-variable case, and in [31], [32] for the multivariate case. The problem of tropical rational function simplification was studied in [33]. Another related problem is approximation in tropical algebra (see, e.g., [34]).

Contribution: This work deals with the problem of tropical polynomial division and its applications in simplifying neural networks. The analysis generalizes previous works [10], [11], [12] to include polynomials with real coefficients. We first introduce a new notion of division and show that there is a unique quotient-remainder pair. Furthermore, we show that the quotient is equal to the convex bi-conjugate of the difference between the dividend and the divisor. Then, the quotient is characterized in terms of the closed convex hull of the union of a finite collection of convex (unbounded) polyhedra. This characterization leads to a simple exact algorithm based on polyhedral geometry. We propose an efficient approximate scheme inspired by an optimization algorithm for convex, piecewise-linear fitting [35]. The approximate algorithm alternates between data clustering and linear programming. Then, we focus on developing algorithms for dividing composite polynomials expressed as the sum of simpler ones. The division results are then applied to simplify neural networks with ReLU activation functions. The resulting neural network has fewer neurons with maxout activations and a reduced total number of parameters. Finally, we present numerical results for the MNIST handwritten digit, SVHN, CIFAR-10 and CIFAR-100 image classification problems. As a baseline for comparison, we use the structured L1 pruning without retraining. The proposed method is very competitive in the large compression regime, that is, in the case where

The authors are with the National Technical University of Athens, School of Electrical and Computer Engineering, Greece.
Author emails: kordonis@central.ntua.gr, maragos@cs.ntua.gr.

The research project was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the “2nd Call for H.F.R.I. Research Projects to support Faculty Members & Researchers” (Project Number:2656, Acronym: TROGEMAL).

there are very few parameters remaining after compression. We also present an additional application on learning Model Predictive Control (MPC), where tropical division is used to approximate the optimal cost-to-go.

The rest of the paper is organized as follows: Section II presents some preliminary algebraic and geometric material. In Section III, we develop some theoretical tools for tropical polynomial division. An exact division algorithm is presented in Section IV, and an approximate algorithm in Section V. Section VI deals with the division of composite tropical polynomials. Finally, VII gives some numerical examples. The proofs of the theoretical results are presented in the Appendix.

II. Preliminaries, Background and notation

Max-plus algebra is a modification of the usual algebra of the real numbers, where the usual summation is substituted by maximum and the usual multiplication is substituted by summation [36], [37], [38], [39]. Particularly, max-plus algebra is defined on the set $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$ with the binary operations "$\vee$" and "$+$". The operation "$+$" is the usual addition. The operation "$\vee$" stands for the maximum, i.e. for $x, y \in \mathbb{R}_{\max}$, it holds $x \vee y = \max\{x, y\}$. For more than two terms we use"$\bigvee$", i.e., $\bigvee_{i\in I} x_i = \sup\{x_i : i \in I\}$. We will also use the '$\wedge$' notation to denote the minimum, i.e., $x \wedge y = \min\{x, y\}$. A unified view of max-plus and several related algebras, was developed in [40], in terms of weighted lattices.

A tropical polynomial is a polynomial in the max-plus algebra. Particularly, a tropical polynomial function [41], [42], [43] is defined as $p : \mathbb{R}^n \to \mathbb{R}_{\max}$ with

$$p(\boldsymbol{x}) = \bigvee_{i=1}^{m_p} (\boldsymbol{a}_i^T \boldsymbol{x} + b_i), \tag{1}$$

where $\boldsymbol{a}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}_{\max}$. In this paper, we will use the terms tropical polynomial and tropical polynomial function interchangeably.

For the tropical polynomial $p(\boldsymbol{x})$, the Newton polytope is defined as

$$\text{Newt}(p) = \text{conv}\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m_p}\}, \tag{2}$$

where "conv" stands for the convex hull, and extended Newton polytope as

$$\text{ENewt}(p) = \text{conv}\{[\boldsymbol{a}_1^T \ b_1]^T, \ldots, [\boldsymbol{a}_{m_p}^T b_{m_p}]^T\}. \tag{3}$$

The values of the polynomial function p depend on the upper hull of the extended Newton polytope [1] (see also [4]). A polyhedron is subset of $\mathbb{R}^n$ which can be represented either as the set of solutions of a finite collection of linear inequalities ($\mathcal{H}-$representation)

$$P = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\},$$

or in terms of a set of vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_l \in \mathbb{R}^n$ and a set of rays $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_s \in \mathbb{R}^n$ as

$$P = \left\{ \sum_{j=1}^{l} \lambda_j \boldsymbol{v}_j + \sum_{j=1}^{s} \mu_j \boldsymbol{w}_j : \lambda_j \geq 0, \sum_{j=1}^{s} \lambda_j = 1, \mu_j \geq 0 \right\}, \tag{4}$$

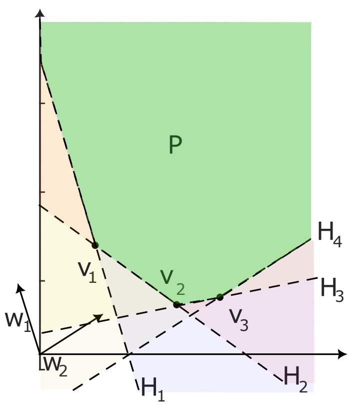

Fig. 1. *Illustration of the Minkovski-Weil theorem. The unbounded polyhedron P (green color in the figure) has two representations: $\mathcal{H}-$reprsentation as the intersection of halfspaces $H_1, \ldots, H_4$ and $\mathcal{V}-$representation generated by vertices v_1, v_2, v_3 and rays w_1, w_2.*

($\mathcal{V}-$representation). The Minkowski-Weyl (resolution) theorem states that each polyhedron admits both representations [44]. Figure 1 illustrates the Minkowski-Weyl resolution of an unbounded polyhedron. Furthermore, there are several well-known algorithms for representation conversion (e.g., [44], Ch. 9). Bounded polyhedra are called polytopes. It is often convenient to describe polytopes in terms of extended representations (e.g., [45]). An extension of a polytope $P \subset \mathbb{R}^n$ is a polyhedron $Q \subset \mathbb{R}^{n'}$, with $n' > n$, along with a linear projection $F : R^{n'} \to \mathbb{R}^n$ such that $F(Q) = P$.

For a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ the epigraph is defined as

$$\text{epi}(f) = \{(\boldsymbol{x}, z) \in \mathbb{R}^{n+1} : z \geq f(\boldsymbol{x})\}.$$

The convex conjugate of f is a function $f^\star : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty, -\infty\}$ given by (see [46])

$$f^\star(\boldsymbol{a}) = \sup\{\boldsymbol{a}^T \boldsymbol{x} - f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^n\}.$$

For a pair of polyhedra $P_1, P_2 \subset \mathbb{R}^n$ the Minkowski sum is defined as

$$P_1 \oplus P_2 = \{x + y : x \in P_1, y \in P_2\}.$$

The following properties hold (see [4])

$$\begin{aligned} \text{Newt}(p_1 + p_2) &= \text{Newt}(p_1) \oplus \text{Newt}(p_2), \\ \text{Newt}(p_1 \vee p_2) &= \text{conv}(\text{Newt}(p_1) \cup \text{Newt}(p_2)). \end{aligned}$$

Similar relations hold for the extended Newton polytopes as well.

III. Tropical Polynomial Division Definition and Existence

We first define the tropical polynomial division.

***Definition** 1:* Let p, d be tropical polynomials. We define the quotient and the remainder of the division of p by d.

(a) A tropical polynomial q is the quotient if

$$p(\boldsymbol{x}) \geq q(\boldsymbol{x}) + d(\boldsymbol{x}), \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^n, \tag{5}$$

and q is the maximum polynomial satisfying this inequality. Particularly, for any tropical polynomial $\tilde{q}$, such that $p(\boldsymbol{x}) \geq \tilde{q}(\boldsymbol{x}) + d(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathbb{R}^n$, it holds $\tilde{q}(\boldsymbol{x}) \leq q(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathbb{R}^n$.

(b) A tropical polynomial r is the remainder of the division of p by d with quotient q if

$$p(\boldsymbol{x}) = (q(\boldsymbol{x}) + d(\boldsymbol{x})) \vee r(\boldsymbol{x}), \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^n, \tag{6}$$

and r is the minimum tropical polynomial satisfying this equality. Particularly, for any tropical polynomial $\tilde{r}$, such that $p(\boldsymbol{x}) = (q(\boldsymbol{x}) + d(\boldsymbol{x})) \vee \tilde{r}(\boldsymbol{x})$, it holds $\tilde{r}(\boldsymbol{x}) \geq r(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathbb{R}^n$.

Observe that the set of tropical polynomials q satisfying (5) is closed under the pointwise maximum. That is, assume that q_1, q_2 are tropical polynomials such that $p(\boldsymbol{x}) \geq d(\boldsymbol{x}) + q_1(\boldsymbol{x})$ and $p(\boldsymbol{x}) \geq d(\boldsymbol{x}) + q_2(\boldsymbol{x})$, then it holds $p(\boldsymbol{x}) \geq d(\boldsymbol{x}) + (q_1(\boldsymbol{x}) \vee q_2(\boldsymbol{x}))$. This property motivates us to determine the monomials

$$q_M^{\boldsymbol{a},b}(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + b,$$

that satisfy (5). For each monomial coefficient $\boldsymbol{a}$, define the maximum value of b that satisfy $p(\boldsymbol{x}) \geq d(\boldsymbol{x}) + q_M^{\boldsymbol{a},b}(\boldsymbol{x})$ as

$$l(\boldsymbol{a}) = \sup\{b : q_M^{\boldsymbol{a},b}(\boldsymbol{x}) + d(\boldsymbol{x}) \leq p(\boldsymbol{x}), \text{for all } x \in \mathbb{R}^n\}. \tag{7}$$

Note that $l(\boldsymbol{a})$ can be written as

$$\begin{aligned} l(\boldsymbol{a}) &= \sup\{b : b \leq p(\boldsymbol{x}) - d(\boldsymbol{x}) - \boldsymbol{a}^T\boldsymbol{x}, \text{for all } x \in \mathbb{R}^n\} \\ &= \inf_{\boldsymbol{x}\in\mathbb{R}^n}\{p(\boldsymbol{x}) - d(\boldsymbol{x}) - \boldsymbol{a}^T\boldsymbol{x}\}. \end{aligned}$$

Denoting by $f(\boldsymbol{x})$ the difference $p(\boldsymbol{x}) - d(\boldsymbol{x})$, we have

$$l(\boldsymbol{a}) = -f^\star(\boldsymbol{x}),$$

where $f^\star$ is the convex conjugate of f. A candidate for the quotient is

$$q(\boldsymbol{x}) = \sup_{\boldsymbol{a}\in\mathbb{R}^n}\{\boldsymbol{a}^T\boldsymbol{x} + l(\boldsymbol{a})\}.$$

Note that q is the bi-conjugate of f. Indeed

$$q(\boldsymbol{x}) = \sup_{\boldsymbol{a}\in\mathbb{R}^n}\{\boldsymbol{a}^T\boldsymbol{x} - f^\star(\boldsymbol{a})\} = f^{\star\star}(\boldsymbol{x}). \tag{8}$$

The following proposition shows that $q(\boldsymbol{x})$ is indeed the quotient of the division.

***Proposition** 1:* For any pair of tropical polynomials p, d there is a quotient-remainder pair. The quotient is given by (8). Furthermore, the polynomial functions of the quotient and remainder are unique.

Proof See Appendix A.

***Remark** 1:* The use of the convex conjugate is closely related to the slope transform [47], [48], [49]. The slope transform was used to derive an analog of frequency response of max-plus and more generally morhological systems.

***Example** 1 (Tropical Polynomial Division in 1D):* Let $p(x) = \max(-2x-1, 1, x+1, 3x-3)$ and $d(x) = \max(x, 2x-1)$. Equivalently, the polynomial functions p and d are written as

$$p(x) = \begin{cases} -2x-1, & \text{if } x \leq -1 \\ 1, & \text{if } -1 < x \leq 0 \\ x+1, & \text{if } 0 < x \leq 2 \\ 3x-3, & \text{if } x \geq 2 \end{cases},$$

$$d(x) = \begin{cases} x, & \text{if } x \leq 1 \\ 2x-1, & \text{if } x > 1 \end{cases}.$$

Thus,

$$f(x) = p(x) - d(x) = \begin{cases} -3x-1, & \text{if } x \leq -1 \\ 1-x, & \text{if } -1 < x \leq 0 \\ 1, & \text{if } 0 < x \leq 1 \\ -x+2, & \text{if } 1 < x \leq 2 \\ x-2, & \text{if } x \geq 2 \end{cases}$$

The plots of p, d and f are given in Figure 2. The difference f is not convex since the slopes $-3, -1, 0, -1, 1$ are not increasing. The quotient q is given as the convex bi-conjugate of f, that is the largest convex function which is less than or equal to f, for all x. Thus, the quotient is given $q(x) = \max(-3x+1, 1-x, -0.5x+1, x-2)$. The sum $d(x)+q(x)$ is equal to $p(x)$, for all x, except $x \in (0, 2)$. Thus, $r(x) = x+1$, that is, the form of $p(x)$, for $x \in (0, 2)$.

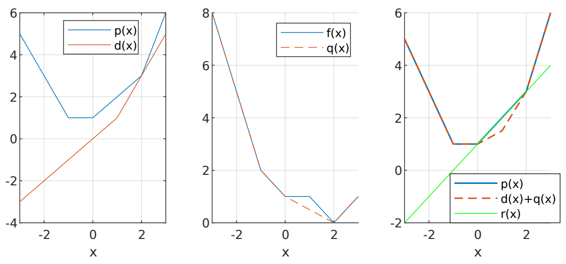

Fig. 2. *This figure refers to Example 1. The left plot presents the dividend p and the divisor d. The middle plot their difference f and the quotient q. The right part shows the dividend p, its approximation d + q and the remainder r.*

We call the division *nontrivial* if $q(\boldsymbol{x}) > -\infty$ and *effective* if $r(\boldsymbol{x}) \neq p(\boldsymbol{x})$, for some $\boldsymbol{x} \in \mathbb{R}^n$. It is easy to see that an effective division is also nontrivial. As shown in the previous example, these notions are not equivalent.

***Proposition** 2:* Assume that p and d are tropical polynomials. If the division of p by d has quotient q and remainder r. Then:

(a) The division of r by d is not effective.

(b) The division of p by q has quotient d and remainder r.

Proof See Appendix B.

***Remark** 2:* For the Euclidean division of a positive integer p by a positive integer d with quotient q and remainder r, the notion of nontrivial division corresponds to $q \neq 0$ and effective division to $r < p$. For the division of positive integers, we observe that these notions are equivalent. Furthermore, the converse of (a) is true. Particularly, if for some $\bar{q}, \bar{r} \in \mathbb{N}$, it holds $p = d\bar{q} + \bar{r}$ and the division of r by d is not effective, then $\bar{q}$ is the quotient and $\bar{r}$ the remainder.

On the other hand, for tropical polynomials, the converse of (a) is not true. Particularly, it is possible that for a pair of tropical polynomials p, d there is another pair q, r such that (6) is satisfied, the division of r by d is not effective (or even is trivial), but q, r is not a quotient-remainder pair. For example consider $p(x) = \max(3x, -3x), d(x) = \max(2x, -2x)$ and

$q(x) = \max(0, x), r(x) = \max(0, -3x)$. It is easy to see that the division of r by d is trivial.

Consider the set

$$C = \{\boldsymbol{c} \in \mathbb{R}^n : \text{Newt}(\boldsymbol{c}^T \boldsymbol{x} + d(\boldsymbol{x})) \subset \text{Newt}(p(\boldsymbol{x}))\}. \quad (9)$$

This set appears also [10], [11], [12] for the case of discrete coefficients. It is not difficult to see that C is a convex polytope. The following proposition shows that the division is non-trivial if and only if $C \neq 0$.

***Proposition** 3:* Let C be the set defined in (9). Then,

(a) C is equal to the domain of function $l(\cdot)$ (given by (7)). That is,

$$C = \textbf{dom}(l(\boldsymbol{a})) = \{\boldsymbol{a} \in \mathbb{R}^n : l(\boldsymbol{a}) > -\infty\}.$$

(b) The division is non-trivial if and only if $C \neq \emptyset$

(c) It the quotient has the form $q(\boldsymbol{x}) = \bigvee_{i=1}^{m_q} (\hat{\boldsymbol{a}}_i^T \boldsymbol{x} + \hat{b}_i)$, then $\hat{\boldsymbol{a}}_i \in C$, for all i.

Proof See Appendix C.

***Corollary** 1:* Assume that the division of a tropical polynomial $p(\boldsymbol{x}) = \bigvee_{i=1}^{m_p} (\boldsymbol{a}_i^T \boldsymbol{x} + b_i)$ by another tropical polynomial $d(\boldsymbol{x}) = \bigvee_{i=1}^{m_d} (\tilde{\boldsymbol{a}}_i^T \boldsymbol{x} + \tilde{b}_i)$ is nontrivial. Then

(i) It holds

$$\text{span}\{\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_{m_d}\} \subset \text{span}\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m_p}\}.$$

(ii) The quotient $q(\boldsymbol{x}) = \bigvee_{i=1}^{m_q} (\hat{\boldsymbol{a}}_i^T \boldsymbol{x} + \hat{b}_i)$ is such that

$$\text{span}\{\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_{m_q}\} \subset \text{span}\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m_p}\}.$$

Proof: See Appendix D.

***Remark** 3:* Corollary 1 can be used to simplify the division of two tropical polynomials. Particularly, assume that

$$\text{span}\{\tilde{\boldsymbol{a}}_1, \ldots, \tilde{\boldsymbol{a}}_{m_d}\} \subset \text{span}\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m_p}\} \subsetneq \mathbb{R}^d.$$

If $\boldsymbol{Q}$ is a matrix the columns of which represent an orthonormal basis of the subspace $\text{span}\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m_p}\}$, then $\boldsymbol{a}_i$ can be expressed in terms of a reduced dimension vector $\boldsymbol{a}_i^r$ as $\boldsymbol{a}_i = \boldsymbol{Q}\boldsymbol{a}_i^r$, and $\tilde{\boldsymbol{a}}_i$ similarly in terms of $\tilde{\boldsymbol{a}}_i^r$ as $\tilde{\boldsymbol{a}}_i = \boldsymbol{Q}\tilde{\boldsymbol{a}}_i^r$. Then, the division of polynomials p and d can be expressed in terms of reduced dimension polynomials p^r, d^r as $p(\boldsymbol{x}) = p^r(\boldsymbol{Q}^T\boldsymbol{x}) = p^r(\tilde{\boldsymbol{x}}), d = d^r(\boldsymbol{Q}^T\boldsymbol{x}) = d^r(\tilde{\boldsymbol{x}})$, where the dimension of $\tilde{\boldsymbol{x}}$ is equal to the rank of $\boldsymbol{Q}$.

This observation is certainly useful when the number of terms of the dividend m_p is less than the space dimension d. It will be also used in Section VI-B.

IV. An Exact Algorithm For Tropical Division

We now present an algorithm to compute the quotient and remainder of a division. The algorithm uses polyhedral computations. Particularly, we use (8), and its equivalent in terms of the epigraphs (see (30) in the appendix) to compute q.

The input of the algorithm is a pair of tropical polynomials

$$p(\boldsymbol{x}) = \bigvee_{i=1}^{m_p} (\boldsymbol{a}_i^T \boldsymbol{x} + b_i), \quad d(\boldsymbol{x}) = \bigvee_{j=1}^{m_d} (\tilde{\boldsymbol{a}}_j^T \boldsymbol{x} + \tilde{b}_j).$$

The first step is to compute a partition of $\mathbb{R}^n$ in polyhedra on which $f(\boldsymbol{x}) = p(\boldsymbol{x}) - d(\boldsymbol{x})$ is linear. To do so, for each i, j, consider the polyhedron on which the terms i, j attain the maximum in p and d respectively.

$$\begin{aligned} P_{i,j} = \{x \in \mathbb{R}^n : \boldsymbol{a}_i^T \boldsymbol{x} + b_i \geq \boldsymbol{a}_{i'}^T \boldsymbol{x} + b_{i'}, \\ \tilde{\boldsymbol{a}}_j^T \boldsymbol{x} + \tilde{b}_j \geq \tilde{\boldsymbol{a}}_{j'}^T \boldsymbol{x} + \tilde{b}_{j'}, i' = 1, \ldots, m_p, j' = 1, \ldots, m_d\}. \end{aligned}$$

Polyhedron $P_{i,j}$ can be compactly written in terms of a matrix $\boldsymbol{A}^{i,j}$ and a vector $\boldsymbol{b}^{i,j}$ as

$$P_{i,j} = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{A}^{i,j}\boldsymbol{x} \geq \boldsymbol{b}^{i,j}\}.$$

Note that some $P_{i,j}$'s can be empty.

The second step is to compute the polyhedra that represent the epigraph of $f(\boldsymbol{x}) = p(\boldsymbol{x}) - d(\boldsymbol{x})$ for each $P_{i,j}$. The epigraphs are given by:

$$\begin{aligned} E_{i,j} = \{(\boldsymbol{x}, z) \in \mathbb{R}^{n+1} : \boldsymbol{A}^{i,j}\boldsymbol{x} \geq \boldsymbol{b}^{i,j}, \\ z \geq (\boldsymbol{a}_i - \tilde{\boldsymbol{a}}_j)^T \boldsymbol{x} + b_i - \tilde{b}_j\}. \end{aligned}$$

For each polyhedron $E_{i,j}$, we compute the corresponding $\mathcal{V}-$representation, consisting of a set of vertices $V_{i,j}$ and a set of rays $R_{i,j}$, satisfying (4). Note that the epigraph of f is given by $\cup_{i,j} E_{i,j}$.

The epigraph of the quotient q is given by the closed convex hull of the epigraph of f (see (30) in the appendix). Next, we consider the union of vertices $V = \cup_{i,j} V_{i,j}$, and rays $R = \cup_{i,j} R_{i,j}$, and consider the polyhedron E generated by V and R (as in (4)). Due to Proposition 1, E is the epigraph of the quotient $q(\boldsymbol{x})$. Then, compute the $\mathcal{H}-$representation of E

$$E = \{(\boldsymbol{x}, z) \in \mathbb{R}^{n+1} : [\boldsymbol{A}^{E,\boldsymbol{x}} \; \boldsymbol{a}^{E,z}][\boldsymbol{x}^T \; z]^T \geq \boldsymbol{b}^E\}, \quad (10)$$

for appropriate matrix $\boldsymbol{A}^{E,\boldsymbol{x}}$ and vectors $\boldsymbol{a}^{E,z}, \boldsymbol{b}^E$. Assume that $E \neq \mathbb{R}^n$. We will prove in Proposition 4 that the components of $\boldsymbol{a}^{E,z}$ are positive. Thus,

$$E = \left\{(\boldsymbol{x}, z) \in \mathbb{R}^{n+1} : z \geq -\frac{1}{[\boldsymbol{a}^{E,z}]_l}[\boldsymbol{A}^{E,\boldsymbol{x}}]_l \boldsymbol{x} + \right. \\ \left. +[\boldsymbol{b}^E]_l/[\boldsymbol{a}^{E,z}]_l, \; l = 1, \ldots, L\right\},$$

where L is the number of rows of $\boldsymbol{A}^{E,\boldsymbol{x}}$. Therefore,

$$q(\boldsymbol{x}) = \bigvee_{l=1}^{L} \left(-\frac{1}{[\boldsymbol{a}^{E,z}]_l}[\boldsymbol{A}^{E,\boldsymbol{x}}]_l \boldsymbol{x} + [\boldsymbol{b}^E]_l/[\boldsymbol{a}^{E,z}]_l\right). \quad (11)$$

The procedure is illustrated in Figure 3.

Having computed the quotient, we formulate the sum $\tilde{p}(\boldsymbol{x}) = d(\boldsymbol{x}) + q(\boldsymbol{x})$. For each $P_{i,j}$ we choose a point $\boldsymbol{x}_{i,j}$ in its relative interior, for example

$$\boldsymbol{x}_{i,j} = \frac{1}{|V_{i,j}|} \sum_{v \in V_{i,j}} v + \sum_{w \in R_{i,j}} w. \quad (12)$$

Let I be the set of indices i such that there is an index j satisfying $p(\boldsymbol{x}_{i,j}) > \tilde{p}(\boldsymbol{x}_{i,j})$. Then,

$$r(\boldsymbol{x}) = \bigvee_{i \in I} (\boldsymbol{a}_i^T \boldsymbol{x} + b_i). \quad (13)$$

The computation of the quotient and the remainder is summarized in Algorithm 1.

***Proposition** 4:* If $E \neq \mathbb{R}^n$, then the output of the algorithm is indeed the quotient and the remainder of the division. If

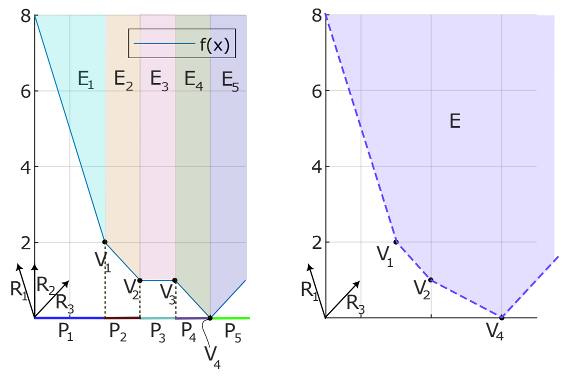

Fig. 3. *Illustration of the algorithm applied to Example 1. Function $f(x) = p(x) - d(x)$ is linear on each of the subsets $P_1, \dots, P_5$. The epigraphs are generated as follows: E_1 by vertex V_1 and rays R_1 and R_2, epigraph E_2 by vertices V_1, V_2 and ray R_2, epigraph E_3 by vertices V_2, V_3 and ray R_2, epigraph E_4 by vertices V_3, V_4 and ray R_2, and finally epigraph E_5 by vertex V_4 and rays R_2, R_3. Their closed convex hull E is generated by vertices V_1, V_2, V_4 and rays R_1, R_3. The quotient $q(x)$ is computed by the $\mathcal{H}-$representation of E.*

Algorithm 1 Exact Polynomial Division

```
0: Input: Tropical Polynomials $p(\boldsymbol{x})$ and $d(\boldsymbol{x})$
1: Compute a partition of $\mathbb{R}^n$ into polyhedra $P_{i,j}$, $i = 1,\dots,m_p$, $j = 1,\dots,m_d$, and their interior points $\boldsymbol{x}_{i,j}$ according to (12)
2: Compute the epigraphs $E_{i,j}$
3: For each epigraph, compute the resolution into a set of vertices $V_{i,j}$ and a set of rays $R_{i,j}$, using a representation conversion algorithm
4: Compute the $\mathcal{H}-$representation of the polyhedron generated by the union of the set of vertices $V = \cup_{i,j} V_{i,j}$, and the union of the set of rays $R = \cup_{i,j} R_{i,j}$, in the form (10)
5: If $L > 0$ then $q(\boldsymbol{x})$ is given by (11). Otherwise, $q(\boldsymbol{x}) = -\infty$.
6: Compute the set of indices $i$ such that there is a $j$ with $p(\boldsymbol{x}_{i,j}) - d(\boldsymbol{x}_{i,j}) - q(\boldsymbol{x}_{i,j}) > 0$. Compute $r(\boldsymbol{x})$ according to (13)
7: Return $q(\boldsymbol{x})$, $r(\boldsymbol{x})$
```

$E = \mathbb{R}^n$, then the quotient is $q(\boldsymbol{x}) = -\infty$, for all $x \in \mathbb{R}^n$ and the remainder is equal to the dividend, i.e., $r = p$.

Proof See Appendix E.

***Remark** 4:* The exact algorithm, as presented, is computationally heavy. A particular bottleneck in its implementation is the calculation of the vertices and rays of each polyhedron, a problem known in the literature as 'vertex enumeration.' The worst-case complexity of vertex enumeration is exponential in the dimension d of the underlying space ([50]). Therefore, we expect that the algorithm will be usable only for small examples.

An alternative method would be to represent the convex hull using Balas' theorem [51]. This would provide a description of the convex hull involving a greater number of variables. Next, we could apply Fourier-Motzkin elimination to reduce the problem to the original dimensionality. While the first step does not have exponential complexity, the worst-case complexity of the second step is exponential.

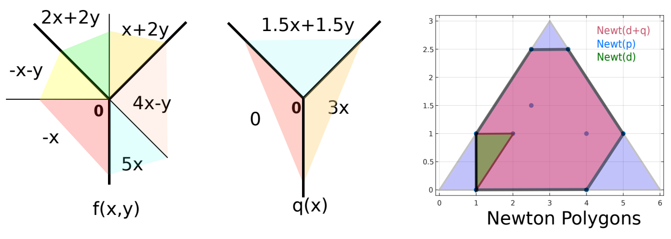

Fig. 4. *The linear regions of $f(x,y)$ and $q(x,y)$. The two functions coincide on the bold half-lines. The figure also illustrates the Newton polygons of $p(x,y)$, $d(x,y)$, and $d(x,y) + q(x,y)$.*

***Example** 2:* Let us divide the tropical polynomial $p(x,y) = \max(0, 3x+3y, 6x)$, by $d(x,y) = \max(x, x+y, 2x+y)$, using Algorithm 1. The quotient is

$$q(x,y) = \max(1.5x + 1.5y, 3x, 0),$$

and $d(x)+q(x)$ is given by $\max(x, x+y, 2.5x+2.5y, 3.5x+2.5y, 5x+y, 4x)$.

The linearity regions of $f(x,y)$ and $q(x)$ are shown in Figure 4. Furthermore, Figure 4 illustrates the Newton polygons for p, d and $d+q$.

V. Approximate Algorithms For Tropical Division

We now present an approximate algorithm for computing the quotient of a tropical division, assuming that the quotient has a predefined maximum number of terms $\tilde{m}_q$. The algorithm mimics the procedure developed in [35].

The quotient has the form $q(\boldsymbol{x}) = \bigvee_{i=1}^{m_q} (\hat{\boldsymbol{a}}_i^T \boldsymbol{x} + \hat{b}_i)$. From the proof of Proposition 1 we know that $q(\boldsymbol{x})$ is the maximum tropical polynomial function, all the terms of which satisfy $\hat{\boldsymbol{a}}_i^T \boldsymbol{x} + \hat{b}_i \le f(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathbb{R}^n$. We will utilize this property to formulate an optimization problem on a set of sample points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathbb{R}^n$. Given the set of sample points and the corresponding set of the values of function $f(\cdot)$, i.e., $f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_N)$, the optimization problem is formulated as:

$$\begin{array}{ll} \underset{\hat{\boldsymbol{a}}_i, \hat{b}_i, i=1,\dots,\tilde{m}_q}{\text{maximize}} & \sum_{j=1}^{N} \bigvee_{i=1}^{\tilde{m}_q} (\hat{\boldsymbol{a}}_i^T \boldsymbol{x}_j + \hat{b}_i) \\ \text{subject to} & \hat{\boldsymbol{a}}_i^T \boldsymbol{x}_j + \hat{b}_i \le f(\boldsymbol{x}_j), \quad \forall i, j \\ & \hat{\boldsymbol{a}}_i \in C, \quad i = 1, \dots, \tilde{m}_q \end{array}, \tag{14}$$

where C is the set defined in (9). Since C is a polytope, the last constraint in (16) can be written as a set of linear inequalities. The details are presented in subsection V-A.

The algorithm alternates between two phases. Phase 1 partitions the data. Particularly, assuming a set of solutions $(\hat{\boldsymbol{a}}_i, b_i)$, $i = 1, \dots, \tilde{m}_q$, we partition the samples $\boldsymbol{x}_j$ into sets $I_1, \dots, I_{\tilde{m}_q}$ with $j \in I_i$, if

$$\hat{\boldsymbol{a}}_i^T \boldsymbol{x}_j + \hat{b}_i \ge \hat{\boldsymbol{a}}_{i'}^T \boldsymbol{x}_j + \hat{b}_{i'}, \quad \text{for all } i'. \tag{15}$$

Phase 2, solves the linear optimization problem:

$$\begin{array}{ll} \underset{\hat{\boldsymbol{a}}_i, \hat{b}_i, i=1,\dots,\tilde{m}_q}{\text{maximize}} & \sum_{i=1}^{\tilde{m}_q} \sum_{j \in I_i} \hat{\boldsymbol{a}}_i^T \boldsymbol{x}_j + \hat{b}_i \\ \text{subject to} & \hat{\boldsymbol{a}}_i^T \boldsymbol{x}_j + \hat{b}_i \le f(\boldsymbol{x}_j), \quad \forall i, j \\ & \hat{\boldsymbol{a}}_i \in C, \quad i = 1, \dots, \tilde{m}_q \end{array} . \tag{16}$$

We then simplify (16) in two ways. First, it may contain redundant inequalities. To remove the redundant inequalities we compute the lower convex hull of the set $\{(\boldsymbol{x}_j, f(\boldsymbol{x}_j)) : j = 1, \dots, N\}$ with respect to its last component, and denote by J the set of indices j, for which $(\boldsymbol{x}_j, f(\boldsymbol{x}_j)$ belongs to the lower convex hull.

Second, problem (16) decomposes into a set of $\tilde{m}_q$ linear programs

$$\begin{array}{ll} \underset{\hat{\boldsymbol{a}}_i, b_i}{\text{maximize}} & \boldsymbol{s}_i^T \hat{\boldsymbol{a}}_i + N_i \hat{b}_i \\ \text{subject to} & [(\boldsymbol{x}_j)^T \ 1] \begin{bmatrix} \hat{\boldsymbol{a}}_i \\ \hat{b}_i \end{bmatrix} \le f(\boldsymbol{x}_j), \quad j \in J \\ & \hat{\boldsymbol{a}}_i \in C \end{array} , \tag{17}$$

where $\boldsymbol{s}_i = \sum_{j \in I_i} \boldsymbol{x}_j$, $N_i = |I_i|$.

The computations are summarized in Algorithm 2.

Algorithm 2 Approximate Polynomial Division

```
0: Input: Tropical Polynomials $p(\boldsymbol{x})$, $d(\boldsymbol{x})$, the degree of the approximate quotient $\tilde{m}_q$, the number of sample points $N$, the maximum number of iterations
1: Compute the inequalities corresponding to set $C$ using the formulas in Appendix V-A
2: Sample the points $\boldsymbol{x}_j$, for $j = 1, \dots, N$, and compute the values of $f(\boldsymbol{x}_j) = p(\boldsymbol{x}_j) - d(\boldsymbol{x}_j)$
3: Compute the lower convex hull of the set $\{(\boldsymbol{x}_j, f(\boldsymbol{x}_j)) : j = 1, \dots, N\}$, and use the samples belonging to it to formulate the first constraint in (16)
4: Initialize the partition $I_1, \dots, I_{\tilde{m}_q}$
5: Solve Problems (17), for all $i$
6: Compute the partitions $I_i$ that satisfy (15)
7: If the number of iterations has not exceeded the maximum number of iterations go to Step 5
8: Return $\hat{q}(\boldsymbol{x}) = \bigvee_{i=1}^{\tilde{m}_q} \hat{\boldsymbol{a}}_i^T \boldsymbol{x} + \hat{b}_i$,
```

We then examine the quality of approximation of $p(\boldsymbol{x})$ by $d(\boldsymbol{x}) + q_t(\boldsymbol{x})$ on the sample set given by

$$e(t) = \sum_{j=1}^{N} [p(\boldsymbol{x}_j) - d(\boldsymbol{x}_j) - q_t(\boldsymbol{x}_j)], \tag{18}$$

where q_t is the quotient computed after t steps of the algorithm. The following proposition shows that the quality of approximation improves as the number of steps increases.

***Proposition** 5:* The quantity $e(t)$ is nonnegative and non-increasing with respect to t.

Proof See Appendix F.

***Remark** 5:* In practice some issues with Algorithm 1 may arise. Some of the classes I_i may end up empty. In this case, we may split another existing class randomly. Furthermore, we expect the algorithm to converge to a local optimum. We may use a multiple start method to avoid bad local optima.

***Remark** 6:* Let us comment about the relationship between Algorithm 2 and [35]. Inspired by [35], Algorithm 2 partitions the data in several clusters. The major difference is that in Algorithm 2 we use the sharper lower linear approximation of each class of data subject to the constraint that this approximation does not exceed any sample point of any class. Thus, instead of alternating between clustering and linear regression, we alternate between clustering and linear programming.

Another related work is [52] which studies the problem of computing a convex under-approximation for a given dataset by a piecewise linear function. The algorithm uses an iterative linear programming technique and has two phases. At phase one of each iteration, it considers a subset of the data set and minimizes the maximum approximation error for this subset, using a piecewise linear function with a number of terms equal to the iteration count. At phase two, it adds to this subset the data point that maximizes the approximation error. Then, it moves to the next iteration.

A. The Linear Constraints for Set C

We next compute a set of linear inequalities describing set C. Let the extreme points of $\text{Newt}(p)$ be $\boldsymbol{a}_{e,1}, \dots, \boldsymbol{a}_{e,k}$. Then, $\hat{\boldsymbol{a}} \in C$ if and only if $\hat{\boldsymbol{a}} + \text{Newt}(d(\boldsymbol{x})) \subset \text{Newt}(p(\boldsymbol{x}))$. That is,

$$\hat{\boldsymbol{a}} + \tilde{\boldsymbol{a}}_j \in \text{conv}\{\boldsymbol{a}_{e,1}, \dots, \boldsymbol{a}_{e,k}\},$$

for all j. Equivalently if there are auxiliary variables $\lambda_{j,l}$, such that

$$\hat{\boldsymbol{a}} + \tilde{\boldsymbol{a}}_j = \sum_{l=1}^{k} \boldsymbol{a}_{e,l} \lambda_{j,l}, \quad \lambda_{j,l} \ge 0, \quad \sum_{l=1}^{k} \lambda_{j,l} = 1$$

for all $j = 1, \dots, m_d$, $l = 1, \dots, k$.

Problem (17) becomes:

$$\begin{array}{ll} \underset{\hat{\boldsymbol{a}}_i, b_i, \lambda}{\text{maximize}} & \boldsymbol{s}_i^T \hat{\boldsymbol{a}}_i + N_i \hat{b}_i \\ \text{subject to} & [\boldsymbol{x}_j^T \ 1] \begin{bmatrix} \hat{\boldsymbol{a}}_i \\ \hat{b}_i \end{bmatrix} \le f(\boldsymbol{x}_j), \quad j \in J \\ & \hat{\boldsymbol{a}}_i + \tilde{\boldsymbol{a}}_j = \sum_{l=1}^{k} \boldsymbol{a}_{e,l} \lambda_{j,l}, \\ & \lambda_{j,l} \ge 0 \\ & \sum_{l=1}^{k} \lambda_{j,l} = 1 \end{array} . \tag{19}$$

***Remark** 7:* In the case where we divide by a monomial, i.e., a polynomial with only a single linear term, the second constraint of (19) simplifies to

$$\hat{\boldsymbol{a}}_i = -\tilde{\boldsymbol{a}} + \sum_{l=1}^{k} \boldsymbol{a}_{e,l} \lambda_l,$$

where we omit the j subscript of λ, since it can take only a single value. Thus, we can eliminate the $\hat{\boldsymbol{a}}_i$ variable and use only the λ variables. Then, we end up with a Linear Programming problem with m_p variables and $N + m_p +$

1 constraints. Note that this formulation will be used for simplifying neural networks in Section VII.

If N is in the order of magnitude of m_p, the linear programming problem can be solved with accuracy ε in $O(m_p^{3.5}\log(1/\varepsilon))$ steps with interior point methods (e.g. [53]). There are m_d such problems, thus the complexity of Phase 2 of the algorithm is $O(m_d \cdot m_p^{3.5}\log(1/\varepsilon))$. Numerical results show that iteration between Phase 1 and Phase 2 converges in a few iterations.

VI. Division for Composite Polynomials

In this section, we present some analytical results and some algorithms for dividing tropical polynomials that are sums or maxima of simpler ones.

A. General Properties

We will use $Q(p,d)$ and $R(p,d)$ to denote the quotient and remainder of the division of a tropical polynomial p by another tropical polynomial d.

***Proposition** 6:* The following inequalities hold

$$Q(p_1+p_2,d) \geq Q(p_1,d)+Q(p_2,d)+d, \tag{20}$$
$$Q(p_1 \vee p_2,d) \geq Q(p_1,d) \vee Q(p_2,d), \tag{21}$$
$$Q(p,d_1+d_2) \geq Q(p,d_1)+Q(p,d_2)-p, \tag{22}$$
$$Q(p,d_1 \vee d_2) \geq Q(p,d_1) \wedge Q(p,d_2). \tag{23}$$

Furthermore, if $R(p_1,d) = R(p_2,d) = -\infty$, then (20),(21),(22) hold as equalities. Finally, if s is a monomial, i.e., it has the form $s(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x}+b$, then

$$Q(p+s,d) = Q(p,d)+s = Q(p,d-s), \tag{24}$$

Proof See Appendix G.

***Remark** 8:* Assume that polynomial d is fixed. Inequality (21) implies that function $Q(\cdot,d)$ is nondecreasing. If $d(\boldsymbol{x}) \geq 0$, for all $\boldsymbol{x}\in\mathbb{R}^n$, then function $Q(\cdot,d)$ is also super-additive.

In the following we call a tropical polynomial described as a summation of other tropical polynomials as composite. Furthermore, we refer to a tropical polynomial in the form 1 as simple[1].

B. Dividing a Composite Polynomial by a Simple

In the following we propose algorithms to approximately divide a composite polynomial $p(\boldsymbol{x}) = \sum_{\nu=1}^N p^\nu(\boldsymbol{x})$ or $p(\boldsymbol{x}) = \bigvee_{\nu=1}^N p^\nu(\boldsymbol{x})$ by another polynomial $d(\boldsymbol{x})$. The computation of the value of the composite polynomial $p(\boldsymbol{x})$ is trivial, provided the values of the simpler polynomials $p^\nu(\boldsymbol{x})$. Thus, the only difficulty in order to apply Algorithm 2 is the computation of the Newton polytope Newt(p).

It is convenient to consider the Newton polytopes Newt(p^ν) in terms of an extended description. Particularly, assume matrices $\boldsymbol{A}^\nu, \boldsymbol{F}^\nu$, and vectors $\boldsymbol{\beta}^\nu$ of appropriate dimensions, such that

$$\text{Newt}(p^\nu) = \{\boldsymbol{a}\in\mathbb{R}^n : \boldsymbol{a} = \boldsymbol{F}^\nu\boldsymbol{\alpha}^\nu, \boldsymbol{A}^\nu\boldsymbol{\alpha}^\nu \leq \boldsymbol{\beta}_\nu\}. \tag{25}$$

[1]These notions refer to the representation of the tropical polynomials, and not to whether or not they can be written as a sum of other polynomials. That is, it is possible that a tropical polynomial with a simple representation (in the form (1)), can be factorized and written in a composite form.

For $p(\boldsymbol{x}) = \sum_{\nu=1}^N p^\nu(\boldsymbol{x})$, the Newton polytope is given by

$$\text{Newt}(p) = \{\boldsymbol{F}^1\boldsymbol{\alpha}^1+\cdots+\boldsymbol{F}^N\boldsymbol{\alpha}^N : \boldsymbol{A}^\nu\boldsymbol{\alpha}^\nu \leq \boldsymbol{\beta}_\nu, \nu=1,\ldots,N\},$$

which corresponds to the Minkowski sum Newt$(p^1)\oplus\cdots\oplus$ Newt(p^N). For $p(\boldsymbol{x}) = \bigvee_{\nu=1}^N p^\nu(\boldsymbol{x})$, the Newton polytope is given by

$$\begin{aligned}\text{Newt}(p) =&\{\boldsymbol{F}^1\boldsymbol{\alpha}^1+\cdots+\boldsymbol{F}^N\boldsymbol{\alpha}^N : \exists\lambda^\nu\geq 0, \nu=1,\ldots,N,\\ &\boldsymbol{A}^\nu\boldsymbol{\alpha}^\nu \leq \boldsymbol{\beta}_\nu\lambda^\nu,\ \sum_{\nu=1}^N\lambda^\nu = 1\}.\end{aligned}$$

C. Neural Networks As Differences of Composite Polynomials

We then present some examples of neural networks represented as the difference of two composite tropical polynomials.

***Example** 3 (Single hidden layer ReLU network):* This example follows [10]. Consider a neural network with n inputs, a single hidden layer with m_p neurons and ReLU activation functions, and a single linear output neuron. The output of the ν-th neuron of the hidden layer is:

$$z_\nu = \max(\boldsymbol{w}_\nu^T\boldsymbol{x}+\beta_\nu, 0)$$

The output y can be written as

$$y = \sum_{\nu=1}^{N_1} w_\nu^{2+}z_\nu - \sum_{\nu=N_1+1}^{N} w_\nu^{2-}z_\nu + \beta^2,$$

where $N_1+N_2 = N$, and N_1, N_2 are the number of neurons that have positive weight and negative weights respectively. Without loss of generality, we assume that the neurons are ordered such that the weights of the first neurons to be positive and the weights of the last negative. The output of the neural network can be expressed as the difference of two tropical polynomials $p_1(\boldsymbol{x})$, $p_2(\boldsymbol{x})$, plus a constant term. Each of these tropical polynomials is written as the sum of simpler ones

$$\begin{aligned}p_1(\boldsymbol{x}) &= \sum_{\nu=1}^{N_1}\max(w_\nu^{2+}\boldsymbol{w}_\nu^T\boldsymbol{x}+w_\nu^{2+}\beta_\nu, 0)\\ &= \sum_{\nu=1}^{N_1}\max((\boldsymbol{a}_1^\nu)^T\boldsymbol{x}+b_\nu, 0)\\ p_2(\boldsymbol{x}) &= \sum_{\nu=N_1+1}^{N}\max(w_\nu^{2-}\boldsymbol{w}_\nu^T\boldsymbol{x}+w_\nu^{2-}\beta_\nu, 0)\\ &= \sum_{\nu=N_1+1}^{N}\max((\boldsymbol{a}_2^\nu)^T\boldsymbol{x}+b_\nu, 0)\end{aligned}$$

Note that p_1, p_2, expressed in their canonical form (1), can have a large number of terms, corresponding to linear regions of tropical polynomial functions p_1, p_2 (for an enumeration of the linear regions see [3], [4]).

The Newton polytope of the simple tropical polynomial $\max((\boldsymbol{a}_1^\nu)^T\boldsymbol{x}+b_\nu, 0)$ is the line segment $[0,\boldsymbol{a}_1^\nu]$ in the d-dimensional space. This polytope is equivalently described in the form (25) as Newt$(p^\nu) = \{\alpha^\nu\boldsymbol{a}_1^\nu : 0\leq\alpha^\nu\leq 1\}$. Thus, the Newton polytope of p_1 is given by

$$\text{Newt}(p_1) = \{\alpha^1\boldsymbol{a}_1^1+\cdots+\boldsymbol{\alpha}^1\boldsymbol{a}_1^{n_1} : 0\leq\alpha^\nu\leq 1\},$$

which is a zonotope (see also [4]).

The constraint $\hat{\boldsymbol{a}}_i \in C$ of problem (17) becomes

$$\hat{\boldsymbol{a}}_i + \tilde{\boldsymbol{a}}_j = \sum_{l=1}^{n} \boldsymbol{a}^l \lambda_{j,l}, \quad 0 \leq \lambda_{j,l} \leq 1.$$

Finally, note that if the number of neurons of the hidden layer N_1, N_2 is smaller than the input dimension n, then the tropical polynomials p_1, p_2 can be expressed in dimensions N_1, N_2.

***Example** 4 (Multi-class to binary classification with a ReLU network):* Assume a neural network with a single ReLU hidden layer with N units and a softmax output layer with K units trained to classify samples to K classes. The equations are the following

$$z_\nu^1 = \max((\boldsymbol{w}_\nu^1)^T x + b_\nu^1, 0), \quad z_k^2 = (\boldsymbol{w}_k^2)^T \boldsymbol{z}^1, \quad y_k = \frac{e^{z_k^2}}{\sum_{k'} e^{z_{k'}^2}}$$

We then assume, that for a given set of samples, we know that the correct class is either i_1 or i_2. The neural network can be simply reduced for binary classification, substituting the last layer with a single neuron with sigmoid activation with output given by

$$y = \sigma((\boldsymbol{w}_{i_1}^2 - \boldsymbol{w}_{i_2}^2)^T \boldsymbol{z}^2 + \beta^2),$$

where $\sigma(\cdot)$ is the sigmoid activation function,, i.e. $\sigma(z) = 1/(1 - e^{-z})$. Note that this transformation corresponds to the application of Bayes' rule with prior probability of $1/2$ for classes i_1, i_2, and likelihoods given by the output of the initial neural network.

The output of the new network can be represented as a difference of two tropical polynomials as in the previous example.

***Example** 5 (Representing a multi-class network as a vector of tropical polynomials):* Consider a neural network with a single-hidden layer having ReLU activations and a linear output layer with K neurons, followed by softmax. Each output (before the softmax) z_k^2 can be expressed as a difference of two tropical polynomials plus a constant term

$$z_k^2 = p_k^+(\boldsymbol{x}) - p_k^-(\boldsymbol{x}) + b_k.$$

The output is given by

$$y_k = \frac{e^{z_k^2}}{\sum_{k'} e^{z_{k'}^2}}.$$

Let us add to each of z_k^2's the sum of all negative polynomials. We get the tropical polynomials

$$\tilde{z}_k^2 = p_k^+(\boldsymbol{x}) + \sum_{k' \neq k} p_{k'}^-(\boldsymbol{x}) + b_k. \tag{26}$$

Then, it is easy to see that

$$y_k = \frac{e^{\tilde{z}_k^2}}{\sum_{k'} e^{\tilde{z}_{k'}^2}}.$$

Finally, denoting by z_i^1, $i = 1, ...M$ the output of the hidden layer before the ReLU, then $\tilde{z}_k^2$ can be written in the form

$$\tilde{z}_k^2 = \sum_{i=1}^{M} \bar{w}_{k,i} \max(z_i^1, 0), \tag{27}$$

for appropriate non-negative constants $\bar{w}_{k,i}$.

***Remark** 9:* In the tropical framework, the difference of two polynomials is considered as a tropical rational function. The addition of all the negative polynomials (denominators) corresponds to making the tropical fractions have the same denominator.

D. Dividing a Vector of Composite Polynomials

In this subsection, we propose a simplified algorithm for dividing a vector of tropical polynomials by the zero polynomial. The motivation comes from Example 5. Consider a set of tropical polynomials in the form (27).

We then describe a simplified version of the the linear optimization part of Algorithm 2. Consider a set of sample points $\boldsymbol{z}^1(1), \ldots, \boldsymbol{z}^1(N)$. Observe that, with this formulation, both constraints (17) are satisfied by any $\hat{\boldsymbol{a}}_k$, such that its $i-$th component $\hat{a}_{i,k}$ satisfies $0 \leq \hat{a}_{i,k} \leq \bar{w}_{i,k}$ and $\hat{b}_k = 0$. Then, the solution of (17), for the $l-th$ term of the division of the $k-$th polynomial is approximated by the optimal solution of

$$\begin{array}{ll} \underset{\hat{\boldsymbol{a}}_k^l}{\text{maximize}} & \boldsymbol{s}_{l,k}^T \hat{\boldsymbol{a}}_k^l \\ \text{subject to} & 0 \leq \hat{a}_{i,k}^l \leq \bar{w}_{i,k}, \quad i = 1, \ldots, M \end{array}. \tag{28}$$

The solution of (28) is given by:

$$\hat{a}_{i,k}^l = \begin{cases} \bar{w}_{i,k} & \text{if } \; s_{i,l,k} > 0 \\ 0 & \text{otherwise} \end{cases}$$

VII. Numerical Results

This section, presents some numerical examples for the tropical division algorithm. First, we present some simple examples to build some intuition for the behaviour of Algorithm 2. Then, some applications of tropical division to the compression of neural networks are presented. We focus on MNIST handwritten and CIFAR-10 datasets.

A. Numerical Examples for Algorithm 2

As a first example, we present the application of Algorithm 2 to the tropical polynomial division of Example 2. We choose 200 sample points distributed according to the normal distribution with unit variance $\mathcal{N}(0, I_2)$. The algorithm converges in two steps and gives an approximate solution[2].

$$\begin{aligned} \hat{q}(x, y) = &\max(1.5038x + 1.4962y + 0.0182, \\ &0.0003x + 0.0288, 3x + 0.0241), \end{aligned}$$

which is very close to the actual quotient (see Example 2). Running again the algorithm with larger sample sizes we conclude that increasing the number of sample points reduces the error.

As a second example we divide a random tropical polynomial with $m_p = 128$ terms in $n = 3$ dimensions with another having $m_d = 2$ terms. First, we run Algorithm 2, for $\tilde{m}_q = 5$, with multiple initial partitions. The evolution of the error for the several runs is shown in Figure 5. The

[2]The code for the numerical experiments is available online at https://github.com/jkordonis/TropicalML

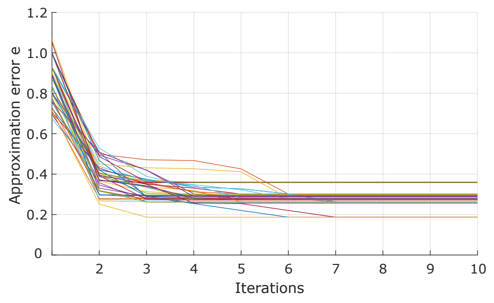

Fig. 5. *The total error* e *for executions of Algorithm 2, for different initial partitions.*

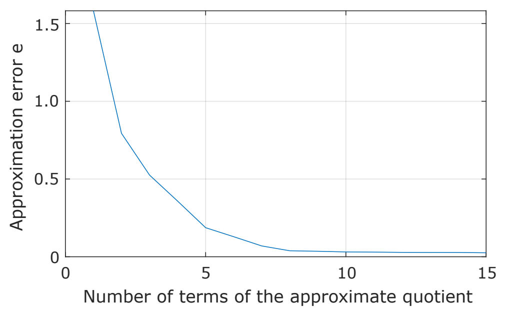

Fig. 6. *The total error* e *for divisions having different number of terms for the approximate quotient*

results illustrate the need for a multi-start method. Indeed, for different random initial partitions the algorithm converges to different local minima. The quality of the approximation e after 10 steps of the algorithm is given in Figure 6 for a varying number of terms $\tilde{m}_q$. This result shows that we may derive a good approximation of the division using a small number of terms (e.g. 8).

B. MNIST Dataset

This example considers the MNIST handwritten digits data set. It consists of 60000 training examples and 10000 test examples of 28×28 gray scale images, that represent digits $0-9$. We start with a single hidden layer neural network with 100 hidden units with ReLU activations and 10 softmax output units. The network is trained in the original training data set using standard techniques (Adam optimizer on cross-entropy loss, batch size of 128). We then design a smaller network discriminating between digits 3 and 5. The input space has $28 \cdot 28 = 784$ dimensions.

We first use the technique described in Example 4 to reduce the output layer of the original network to a single neuron. Then, using the technique described in Example 3, we express the output (before the sigmoid) as the difference of two tropical polynomials. These polynomials are expressed as the summation of 100 terms in the form $\max(\boldsymbol{\alpha}_l^\nu \boldsymbol{x}, 0)$, where $l = 1, 2$. We then apply a reduced QR decomposition to the matrices $\boldsymbol{A}_1 = [\boldsymbol{\alpha}_1^1 \ \ldots, \ \boldsymbol{\alpha}_1^{100}]$ and $\boldsymbol{A}_2 = [\boldsymbol{\alpha}_2^1 \ \ldots, \ \boldsymbol{\alpha}_2^{100}]$, writing them as $\boldsymbol{A}_1 = \boldsymbol{Q}_1 \boldsymbol{A}_1^r$ and $\boldsymbol{A}_2 = \boldsymbol{Q}_2 \boldsymbol{A}_2^r$. The $\boldsymbol{A}_l^r$ matrices have as columns $\boldsymbol{\alpha}_l^{r,\nu}$, i.e., an expression of the vectors $\boldsymbol{\alpha}_l^\nu$ in the subspace of their span. For each polynomial the input has been transformed as $\boldsymbol{x}_l = \boldsymbol{Q}_l^T \boldsymbol{x}$ (recall Remark 3).

We divide each of these polynomials with the zero polynomial $d = 0$, applying the Algorithm 2, with the modification of Section VI-B, and using as sample points the first 200 training points. The outcome of this computation is the approximate quotients q_1 and q_2, in the form $q_l(\boldsymbol{x}_l) = \bigvee_{i=1}^{\tilde{m}_q} (\hat{\boldsymbol{a}}_i^l \boldsymbol{x}_l + \hat{b}_i^l)$. Then, the output of the original network can be approximated as

$$y^{\text{appr}} = \sigma(q_1(\boldsymbol{Q}_1^T \boldsymbol{x}) - q_2(\boldsymbol{Q}_2^T \boldsymbol{x}) + \beta^2)$$

The computation of the approximate output corresponds to a neural network an input layer, a hidden layer with two maxout units, a linear layer with a single unit (computing the difference of the two maxout units) and an output layer with a single sigmoid unit.

Table I presents the error rate on the test set of the original neural network, as well as the computed maxout networks, where each maxout unit has 3, 5, and 10 terms respectively. It also presents the percentage of parameters that remain after the compression. As baselines for comparison, we use structured L1 pruning without retraining and structured SNIP (see e.g. [54], [55]). The first method deletes the neurons having weights with small L1 norm, and the second computes the sensitivity of the total loss with respect to the presence of each neuron. We compare reduced neural networks with the same number of parameters. We present two results, the binary comparison of digits 3 and 5 and an average of the $45 = 10 \cdot 9/2$ different binary comparisons.

Network.	Orig. 2 cl.	$\tilde{m}_q = 10$	$\tilde{m}_q = 5$	$\tilde{m}_q = 3$
Err. L1 Avg.	0.35± 0.25%	14.4±8.87%	26.95 ± 12.24%	34.61± 12.5%
Err. Div. Avg.	0.35± 0.25%	**0.66±0.50%**	**0.83± 0.70%**	**1.12±1.21%**
Err. L1 3-5	1.05%	16.61%	25.08%	36.44%
Err. Div. 3-5	1.05%	**1.57%**	**2.00%**	**2.47%**
Err. SNIP. 3-5	1.05%	4.63%	16.04%	25.87%
# Param.	77001	15381	7691	4615
% of Param. Remaining	100%	20%	10%	6%

TABLE I

Results for the MNIST dataset. Error rate of the original network and the reduced networks. Err. Div. represents the error rate for the reduced networks obtained using the tropical division algorithm and Err. L1 the error rate of with the L1 structured pruning algorithm. The error rates for the 3-5 binary comparison and the average over the 45 binary comparisons are presented.

***Remark** 10:* It is worth noting substituting $\boldsymbol{x}_l$ into q_l, we get

$$q_1(\boldsymbol{Q}_1^T \boldsymbol{x}) = \bigvee_{i=1}^{\tilde{m}_q} (\hat{\boldsymbol{a}}_i^l Q_l^T \boldsymbol{x} + \hat{b}_i^l).$$

Thus we do not need to store Q_1, Q_2 but only the vectors $\hat{\boldsymbol{a}}_i^1 Q_l^T$, and the scalars $\hat{b}_i^l$, β^2, for $l = 1, 2$ and $i = 1, \ldots, m_q$.

***Remark** 11:* Let us note that for computing the reduced networks, we used only the first 200 samples of the training data.

C. CIFAR-10 Dataset-Binary

This example concerns CIFAR-10 dataset which contains small (32×32) color images. The training set consists of 50000 images of 10 classes (5000 samples for each class) and the test set of 10000 images (1000 samples per class). We first train a VGG-like network [56], having three blocks consisting of two convolution layers with ReLU activation followed by a 2×2 max-pooling layer with padding, a dense layer with 1024 neurons, and an output layer with 10 neurons. The convolution layers in the first, second, and third block have 32, 64, and 128 filters, respectively. The network was trained using standard techniques (Adam optimizer for the cross-entropy loss, batch normalization and dropout (0.2) for regularization and data augmentation (shift and horizontal flip)). Let us note that the input of the dense layer has dimension $4\cdot 4\cdot 128 = 2048$.

We then design a simplified neural network discriminating between pairs of classes (e.g., "Automobile" and "Truck"). To do so, we approximate the dense layer with 1024 neurons using the techniques of the previous section. Particularly, using the technique of Example 4 we describe the output of the neural network as the difference of two tropical polynomials in 2048 dimensions. These polynomials are then divided by the zero polynomial applying the Algorithm 2, with the modification of Section VI-B, and using as sample points the first 200 training points.

Table II presents the error rate on the test set when there are 3, and 5 maxout units on the dense layer. Three tests results are presented: on pairs Automobile-Truck (A-T), Cat-Dog (C-D) and the average of the 45=$9\cdot 10/2$ possible binary comparisons. The results obtained with the tropical division algorithm are compared with the L1 structured pruning. Table II also presents the number of parameters of the dense layer of the networks. We observe that the dense layer can be compressed to less than 1% of its original size with minimum performance loss.

Network	Orig. 2 classes	$\tilde{m}_q = 5$	$\tilde{m}_q = 3$
Err. L1 A-T	4.04%	25.8%	46.45%
Err. Div. A-T	4.05%	**4.6%**	**4.4%**
Err. L1 C-D	13.7%	49.3%	50%
Err. Div. C-D	13.7%	**15.4%**	**16.85%**
Err. L1 Avg.	2.74±2.57%	43.45±10.57%	43.62±10.61%
Err. Div. Avg.	2.74±2.57%	**3.85±3.16%**	**4.06±3.37%**
# Param.	$2.1\cdot 10^6$	$2\cdot 10^4$	$1.2\cdot 10^4$
% of Param. Remaining	100%	0.95%	0.57%

TABLE II

Results for the CIFAR-10 dataset. Error rate of the original network, the reduced maxout networks and the reduced networks obtained using the L1 structured pruning algorithm. We present the Automobile vs Truck case, the Cat vs Dog case, and the average of the $45 = 10\cdot 9/2$ different binary comparisons. The results are indexed using A-T, C-D, and Avg.

***Remark** 12:* The proposed scheme is very competitive for the large compression regime, that is, in the case where there are very few parameters remaining. Furthermore, it can be performed assuming access to a small portion of the training data (500 out of 60000).

D. CIFAR-10 Multiclass

We then compress the multi-class neural network trained in the previous subsection for the CIFAR-10 dataset. As the input to the compression algorithm, we consider the output of

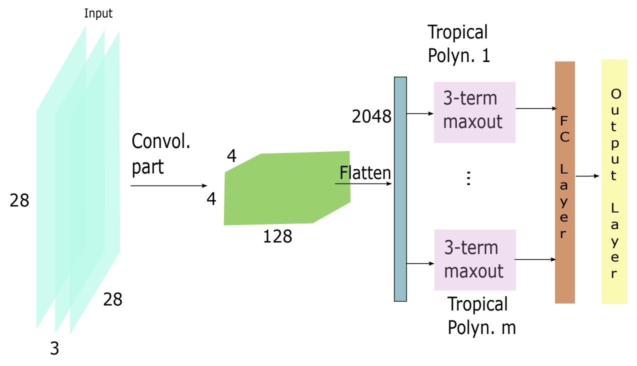

Fig. 7. *The structure of the proposed reduced network. Each pink box consists of linear neurons obtained by successive implementations of the division algorithm.*

the convolutional part of the neural network with dimension 2048.

We first represent the logits (the output before the softmax) as a vector of tropical polynomials using (26) in Example 5. We then simplify each of the tropical polynomials using the simplified algorithm of Section VI-D and a random sample of 500 training points. The results are summarized in Table III. Additionally, we present an improved prediction scheme, where the $K\times\tilde{m}_q$ values $(\hat{\boldsymbol{a}}_k^l)^T\boldsymbol{z}^1$, with $k = 1,\ldots,K$ and $l = 1,\ldots,\tilde{m}_q$ are fed to a small single hidden layer ReLU neural network with 100 units. We refer to this small neural network as head.

Network	Original	$\tilde{m}_q = 7$	$\tilde{m}_q = 5$	$\tilde{m}_q = 3$
Error L1	14.18%	45%	57.3%	71.7%
Error Div.	14.18%	41±1.8 %	40.1 ± 1.7%	42.5 ± 2%
Err. Div. Improved	14.18%	**26.6± 0.3** %	**26.4** ± **0.3%**	**28.1** ± **0.1**%
# Param.	$2.1\cdot 10^6$	$1.4\cdot 10^5$	10^5	$6.6\cdot 10^4$
% of Param. Remaining	100%	7.32%	5.24%	3.16%

TABLE III

Results for the compression on the CIFAR-10 dataset. Average error rates for multiple executions. Comparison of the original division algorithm, the improved division algorithm and the L1 pruning. The number of parameters include also the parameters of the head.

Note that the additional parameters for the improved prediction are very few. Indeed for the case of $\hat{m}_q = 3, 5, 7$ we have 4110, 6110 and 8110 parameters respectively.

We observe again that the division algorithm works well on the large compression regime. Furthermore, in this example there is no improvement when we use more terms in the quotient polynomials. This is probably due to stacking in local optima.

We then exploit the good behaviour of the binary classification algorithm and apply it to the multi-class problem. To do so, we use a subset of the tropical polynomials obtained for binary classification. In each binary comparison, the division algorithm gives two tropical polynomials and the $9\times 10/2 = 45$ binary comparisons give a total of 90 tropical polynomials. The value of a subset of the tropical polynomials is fed to a single hidden layer network with ReLU activations, as described in Figure 7.

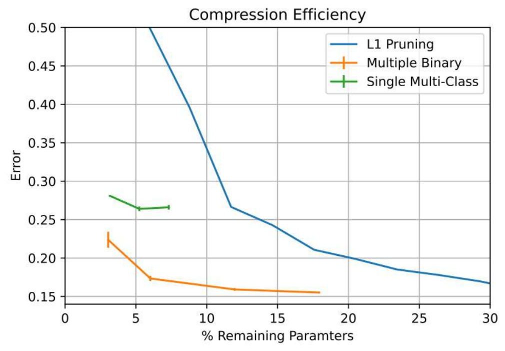

Fig. 8. *The error rate of the compressed neural networks obtained using the multiple division algorithm. The blue and yellow lines use divisions with $\tilde{m}_q = 3$ and $\tilde{m}_q = 5$, respectively. The horizontal axis represents the percentage of the remaining parameters. Finally, the green line corresponds to using random vectors instead of the quotients.*

Figure 8 presents the error of the compressed networks obtained using the multiple binary division algorithm of the previous paragraph. To produce these results we used a random set of m tropical polynomials with $m = 10, 20, 40, 60$.

We observe that the proposed method works better than L1-structured pruning, when the number of remaining parameters is small.

Remark *13:* Note that in all the examples, the tropical division compression algorithms used only a small subset of the training data. This can be useful when simplifying a network where only parts of the training dataset are available.

E. SVHN Dataset

In this section, we consider the SVHN (street view home number) dataset consisting of 32×32 colored images. The training set has 73257 images and the test set of 26032 images, and there are 10 classes. For this dataset, we train a VGG-like convolutional neural network similar to the one described in the previous section. The training was done with standard methods (Adam with learning rate 0.001, dropout 0.5, for 15 epochs).

We then run the multiple binary division algorithm of the previous paragraph. The results are presented in Figure 9.a.

F. CIFAR-100 Dataset

In this section, we consider CIFAR-100 dataset which consists of 60000 small colored images. The training set has

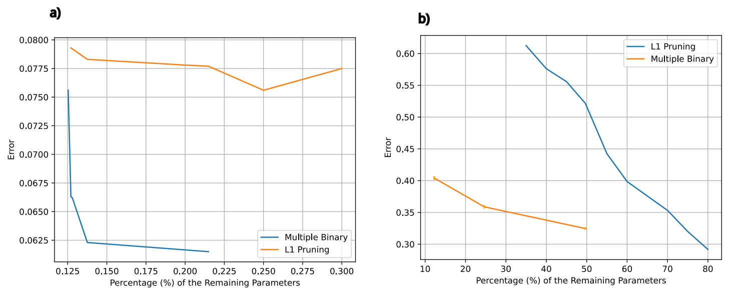

Fig. 9. *The error rate of the compressed neural networks obtained using the multiple division algorithm. The horizontal axis represents the percentage of the remaining parameters. The blue line corresponds to L1 structured pruning, and the orange line to the multiple binary division technique. Part a) presents the results for the SVHN dataset and part b) the results for CIFAR-100.*

50000 samples of 100 classes (500 samples per class) and 10000 test images. We train a convolutional neural network consisting of two initial convolutional layers with 64 and 128 filters, respectively. This is followed by a residual block with two convolutional layers, each containing 128 filters. The network then continues with two additional convolutional layers with 256 and 512 filters, respectively. A second residual block follows, consisting of two convolutional layers with 512 filters. The network includes also a fully connected linear layer with 512 neurons and an output layer. The vast majority of the parameters of this network are in the last residual block having 512 kernels in each layer. The network is trained using standard techniques (Adam optimizer with weight decay, cross-entropy loss, batch normalization and dropout 0.2 for regularization and data augmentation (random crop and horizontal flipping)).

We then use the multiple binary division technique, described in the previous subsection, to simplify the network. The results are presented in Figure 9.b. Again the proposed method is competitive in the large compression regime.

G. Application to Learning Model Predictive Control

In this section, we present an application of tropical division in Learning Model Predictive Control (LMPC) [57], [58], [59], (see also [60]). Let us first describe briefly the usual MPC scheme. Consider a linear system

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t,$$

and the optimization of a long-horizon objective

$$J(\boldsymbol{x}_0) = \sum_{t=0}^{T} g(\boldsymbol{x}_t, \boldsymbol{u}_t),$$

where g is a convex function. At each time step k, MPC measures the state $\boldsymbol{x}_k$ and solves a short-horizon optimization problem

$$\begin{aligned} \text{minimize} \quad & \tilde{J}(\boldsymbol{x}_k) = q(\boldsymbol{x}_{k|k+T_s}) + \sum_{t=k}^{T_s+k-1} g(\boldsymbol{x}_{k|t}, \boldsymbol{u}_{k|t}) \\ \text{subject to} \quad & \boldsymbol{x}_{k|t+1} = \boldsymbol{A}\boldsymbol{x}_{k|t} + \boldsymbol{B}\boldsymbol{u}_{k|t} \\ & \boldsymbol{x}_{k|k} = \boldsymbol{x}_k \end{aligned}, \qquad (29)$$

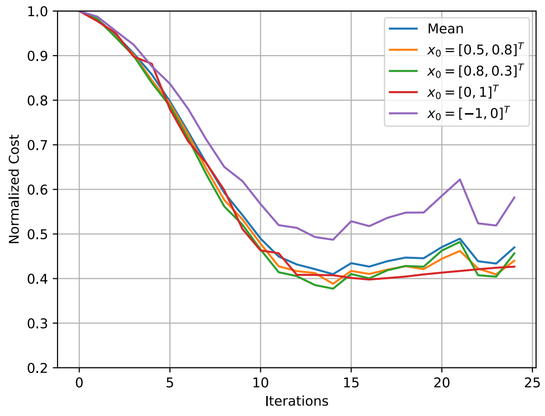

Fig. 10. *The normalized cost for four different initial conditions, as learning progresses.*

where T_s is a smaller horizon. Then, it applies the first computed input $\boldsymbol{u}_k = \boldsymbol{u}_{k|k}$ to the system and proceeds to measure the next state $\boldsymbol{x}_{k+1}$. The process starts then again from step $k+1$.

The LMPC approach learns a function $q(\boldsymbol{x}_{k|k+T_s})$ that approximates the minimum cost-to-go function, defined as the minimum value of $\sum_{t=T_s+k}^{T} g(\boldsymbol{x}_{k|t}, \boldsymbol{u}_{k|t})$. To do so, it employs an iterative scheme. It starts with an initial estimate $q^0(\boldsymbol{x})$ of the optimal cost-to-go. Starting from several values of the initial state $\boldsymbol{x}_0^j$, with $j = 1, \dots, N$, it evaluates the realized costs $J^1, \dots, J^N$, under the MPC policy. This creates a dataset for estimating the cost-to-go. This dataset is used to generate an update $q^1(\boldsymbol{x})$ of the estimate of the optimal cost-to-go. The process is repeated several times until convergence.

It can be shown that under the aforementioned assumptions, the optimal cost-to-go is a convex function. We then present an example, where Algorithm 2 is used to approximate the optimal cost-to-go function.

***Example** 6:* Assume that the state evolution matrices are

$$\boldsymbol{A} = \begin{bmatrix} 0.75 & 0.5 \\ 0.4 & 0.9 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix},$$

and the cost is given by

$$g(\boldsymbol{x}, u) = u^2 + \|\boldsymbol{x}\| + \|\boldsymbol{x}\|^3.$$

The time horizon T_s is 2 and the original horizon T is 50. For each iteration we compute the cost-to-go of 50 random initial conditions under the MPC control scheme, using the old value of q. Then, we run Algorithm 2 (dropping the last constraint of (17)) using $\tilde{m}_q = 4$ terms, to update the estimate of q, and iterate. Figure 10 shows the total cost under the MPC controller as learning progresses, for four different initial conditions.

***Remark** 14:* Since we use tropical division, that is we seek for the best convex under-approximation of the data, there is no need to discard the samples of previous iterations when we try to learn the *minimum* cost-to-go.

***Remark** 15:* The optimization (29) should run online, and thus the use of a simple form for q is important.

VIII. Conclusion

This work proposed a new framework for tropical polynomial division and its application to neural networks. We showed the existence of a unique quotient-remainder pair and characterized the quotient as the convex bi-conjugate of the difference of the dividend from the divisor. This characterization led to an exact algorithm based on polyhedral geometry. We also proposed an approximate algorithm based on the alternation between clustering and linear programming. We then focused on dividing composite tropical polynomials and proposed a modified algorithm. Finally, we applied tropical polynomial division techniques to simplify neural networks with ReLU activations. The resulting neural networks have a maxout activation layer. The results are promising and compare favorably with a simple baseline (L1 pruning).

There are several directions for future research. First, we may combine L1 regularization to the problem (19) to induce sparse solutions and improve further neural network compression. Another direction is to study compression problems for neural networks with many outputs, using a single division. A possible tool in this direction is the Cayley trick [61], [62]. Another direction for future research is to explore simplifying transformer architectures (e.g. [63]) using concepts from tropical division Finally, the study of alternative optimization algorithms is of certain interest.

Acknowledgment

The authors are grateful to Dr. George Retsinas for his valuable suggestions and comments.

Appendix

A. Proof of Proposition 1

It is not difficult to see that if $\tilde{q}(\boldsymbol{x}) = \bigvee_{i=1}^{m_{\tilde{q}}} (\tilde{\boldsymbol{a}}_i^T \boldsymbol{x} + \tilde{b}_i)$, and $\tilde{q}$ satisfies (5), then $\tilde{q}(\boldsymbol{x}) \leq q(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathbb{R}^n$. Indeed, since

$$\tilde{\boldsymbol{a}}_i^T \boldsymbol{x} + \tilde{b}_i \leq \tilde{q}(\boldsymbol{x}) \leq p(\boldsymbol{x}) - d(\boldsymbol{x}),$$

it holds $\tilde{b}_i \leq l(\tilde{\boldsymbol{a}}_i)$. Thus,

$$\tilde{\boldsymbol{a}}_i^T \boldsymbol{x} + \tilde{b}_i \leq \tilde{\boldsymbol{a}}_i^T \boldsymbol{x} + l(\boldsymbol{a}_i) \leq q(\boldsymbol{x}).$$

Hence, $\tilde{q}(\boldsymbol{x}) \leq q(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathbb{R}^n$.

To prove that q is a quotient, it remains to show that it is a tropical polynomial. The epigraph of q is the closed convex hull of the epigraph of f [46]

$$\text{epi}(q) = \overline{\text{conv}(\text{epi}(f))}. \tag{30}$$

On the other hand, f is a piecewise linear function. Thus, its epigraph is a union of a finite number of convex polyhedra. Indeed if $A_1, \dots, A_{m_f}$ its linear regions (i.e., intersections of linear regions of p and d) then A_i, $i = 1, \dots, m_f$ are convex polyhedra and

$$\text{epi}(f) = \bigcup_{i=1}^{m_f} \{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} : \boldsymbol{x} \in A_i, t \geq f(\boldsymbol{x})\}.$$

Observe that epi(q) is a convex polyhedron and q is a piecewise linear convex function. Thus, q is a tropical polynomial. From the definition it is obvious that the quotient is unique.

Let q be the quotient and assume that for some tropical polynomial $\tilde{r}$ it holds

$$p(\boldsymbol{x}) = (q(\boldsymbol{x}) + d(\boldsymbol{x})) \vee \tilde{r}(\boldsymbol{x}), \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^n.$$

Assume that for some point $\boldsymbol{x}_0$ it holds $p(\boldsymbol{x}_0) > d(\boldsymbol{x}_0)+q(\boldsymbol{x}_0)$ and that p has the linear form $\boldsymbol{a}_{i_0}^T\boldsymbol{x} + b_{i_0}$ in a neighborhood of $\boldsymbol{x}_0$. Then, $\tilde{r}(\boldsymbol{x}) = p(\boldsymbol{x})$ in this neighborhood. Furthermore, since $\tilde{r}$ is convex, we have

$$\tilde{r}(\boldsymbol{x}) \geq \bigvee_{i\in I} \boldsymbol{a}_i\boldsymbol{x} + b_i = r(\boldsymbol{x}),$$

where $I = \{i \in \{1,\dots,m_p\} : \exists \boldsymbol{x} \in \mathbb{R}^n \text{ with } p(\boldsymbol{x}) > d(\boldsymbol{x}) + q(\boldsymbol{x}), \text{ and } p(\boldsymbol{x}) = \boldsymbol{a}_i\boldsymbol{x} + b_i\}$. It is not difficult to see that r is a remainder.

B. *Proof of Proposition 2*

a) Let $\bar{q}$ and $\bar{r}$ be the quotient and the remainder of the division of r by d. Then

$$r(\boldsymbol{x}) = (d(\boldsymbol{x}) + \bar{q}(\boldsymbol{x})) \vee \bar{r}(\boldsymbol{x}) \tag{31}$$

Combining with

$$p(\boldsymbol{x}) = (d(\boldsymbol{x}) + q(\boldsymbol{x})) \vee r(\boldsymbol{x}),$$

we get

$$\begin{aligned} p(\boldsymbol{x}) &= (d(\boldsymbol{x}) + q(\boldsymbol{x})) \vee (d(\boldsymbol{x}) + \bar{q}(\boldsymbol{x})) \vee \bar{r}(\boldsymbol{x}) \\ p(\boldsymbol{x}) &= (d(\boldsymbol{x}) + q(\boldsymbol{x}) \vee \bar{q}(\boldsymbol{x})) \vee \bar{r}(\boldsymbol{x}). \end{aligned} \tag{32}$$

Equation (31) implies that $\bar{r}(\boldsymbol{x}) \leq r(\boldsymbol{x})$. This fact combined with the fact that $q(\boldsymbol{x})$ is the quotient of the division of $p(\boldsymbol{x})$ by $d(\boldsymbol{x})$ and (32) implies that $\bar{q}(\boldsymbol{x}) \vee q(\boldsymbol{x}) = q(\boldsymbol{x})$. Hence, since $r(\boldsymbol{x})$ is the remainder of the division of $p(\boldsymbol{x})$ by $d(\boldsymbol{x})$ it holds $r(\boldsymbol{x}) \leq \bar{r}(\boldsymbol{x})$. Hence, $r(\boldsymbol{x}) = \bar{r}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathbb{R}^n$

b) The proof is trivial.

C. *Proof of Proposition 3*

Assume $p(\boldsymbol{x})$ is given by (1) and $d(\boldsymbol{x})$ by

$$d(\boldsymbol{x}) = \bigvee_{i=1}^{m_d} \tilde{\boldsymbol{a}}_i\boldsymbol{x} + \tilde{b}_i.$$

(a) We first show that $C \subset \mathbf{dom}(l(\boldsymbol{a}))$. Assume that $\boldsymbol{a} \in C$. That is $\boldsymbol{a} + \tilde{\boldsymbol{a}}_i \in \text{Newt}(p)$ for all $i = 1,\dots,m_d$. Then, there are $\lambda_1,\dots,\lambda_{m_d} \geq 0$ with $\sum_{i=1}^{m_p} \lambda_1 = 1$ such that

$$\boldsymbol{a} + \tilde{\boldsymbol{a}}_i = \sum_{i=1}^{n_p} \lambda_i\boldsymbol{a}_i.$$

Thus,

$$(\boldsymbol{a} + \tilde{\boldsymbol{a}}_i)^T\boldsymbol{x} \leq \bigvee_{i=1}^{n_p} \boldsymbol{a}_i\boldsymbol{x} \leq p(\boldsymbol{x}) - \bigwedge_{i=1}^{n_p} b_i.$$

Furthermore,

$$\tilde{\boldsymbol{a}}_i\boldsymbol{x} + \tilde{b}_i - \tilde{b}_i \geq \tilde{\boldsymbol{a}}_i\boldsymbol{x} + \tilde{b}_i - \bigvee_{j=1}^{m_d} \tilde{b}_j.$$

Combining the last two equations we have

$$p(\boldsymbol{x}) - d(\boldsymbol{x}) - \boldsymbol{a}^T\boldsymbol{x} \geq \bigwedge_{i=1}^{n_p} b_i - \bigvee_{j=1}^{m_d} \tilde{b}_j > -\infty.$$

Hence, $\boldsymbol{a} \in \mathbf{dom}(l)$.

Conversely assume that $\boldsymbol{a} \in \mathbf{dom}(l)$ but $\boldsymbol{a} \notin C$. Then, there is a $\tilde{\boldsymbol{a}}_i$ such that $\boldsymbol{a} + \boldsymbol{a}_i \notin \text{Newt}(p)$. Furthermore, since $\text{Newt}(p)$ is a convex set, there is a vector $\boldsymbol{v}$ such that $(\boldsymbol{a} + \boldsymbol{a}_i)^T\boldsymbol{v} > \boldsymbol{a}_i^T\boldsymbol{v}$ for all $i = 1,\dots,n_p$. Hence,

$$\begin{aligned} l(\boldsymbol{a}) &= \inf_{\boldsymbol{x}\in\mathbb{R}^n} \{p(\boldsymbol{x}) - d(\boldsymbol{x}) - \boldsymbol{a}^T\boldsymbol{x}\} \\ &\leq \lim_{\lambda\to-\infty} p(\lambda\boldsymbol{v}) - d(\lambda\boldsymbol{v}) - \boldsymbol{a}^T(\lambda\boldsymbol{v}) = -\infty \end{aligned}$$

Thus, $\boldsymbol{a} \notin \mathbf{dom}(l)$, which is a contradiction.

(b) Since

$$q(\boldsymbol{x}) = \sup_{\boldsymbol{a}\in\mathbb{R}^n} \{\boldsymbol{a}^T\boldsymbol{x} - f^\star(\boldsymbol{a})\}$$

It holds

$$q(\boldsymbol{x}) = -\infty \Leftrightarrow l(\boldsymbol{a}) = -\infty, \text{ for all } \boldsymbol{a} \Leftrightarrow C = \mathbf{dom}(l) = \emptyset$$

(c) If for some i it holds $\hat{\boldsymbol{a}}_i \notin C$, then $l(\hat{\boldsymbol{a}}_i) = -\infty$. That is, for all b there is an $\boldsymbol{x} \in \mathbb{R}^n$ with $\hat{\boldsymbol{a}}_i^T\boldsymbol{x} + b > f(\boldsymbol{x})$. This is is particularly true for $b = \hat{b}_i$, and thus $q(\boldsymbol{x})$ is not a quotient.

D. *Proof of Corollary 1*

The proof of the first part is immediate from Proposition 3. To contradict, assume that the inclusion in (ii) is not true. Then, there exists a vector $\boldsymbol{x} \in \text{span}\{\boldsymbol{a}_1,\dots,\boldsymbol{a}_{m_p}\}^\perp$ but $\boldsymbol{x} \notin \text{span}\{\hat{\boldsymbol{a}}_1,\dots,\hat{\boldsymbol{a}}_{m_q}\}^\perp$. Thus, there is an index i_0 such that $\hat{\boldsymbol{a}}_{i_0}^T\boldsymbol{x} \neq 0$. Without loss of generality assume that $\hat{\boldsymbol{a}}_{i_0}^T\boldsymbol{x} > 0$ (if it is negative, use $-\boldsymbol{x}$ in the place of $\boldsymbol{x}$). Using (i) we have $\lim_{\lambda\to\infty}(q(\lambda\boldsymbol{x}) + d(\lambda\boldsymbol{x})) = \infty$, but $\lim_{\lambda\to\infty} p(\lambda\boldsymbol{x}) < \infty$. Hence, there is a λ such that $p(\lambda\boldsymbol{x}) < q(\lambda\boldsymbol{x}) + d(\lambda\boldsymbol{x})$, which contradicts the fact that q is the quotient.

E. *Proof of Proposition 4*

First we need to show that all the components of $\boldsymbol{a}^{E,z}$ are positive. Due to the convexity of E, either

$$\inf\{z : (\boldsymbol{x}, z) \in E\} = -\infty, \text{ for all } \boldsymbol{x} \in \mathbb{R}^n,$$

or

$$\inf\{z : (\boldsymbol{x}, z) \in E\} > -\infty, \text{ for all } \boldsymbol{x} \in \mathbb{R}^n.$$

If $E = \mathbb{R}^{n+1}$, then there are no non-trivial constraints, i.e., $L = 0$. In the following assume that $L > 0$, that is there are some nontrivial constraints, and $E \neq \mathbb{R}^{n+1}$. From the definition of E, for all $\boldsymbol{x} \in \mathbb{R}^n$, it holds $\inf\{z : (\boldsymbol{x}, z) \in E\} < \infty$, and $\sup\{z : (\boldsymbol{x}, z) \in E\} = \infty$.

Note that it is not possible to have a constraint in (10) in the form $[\boldsymbol{A}^{E,\boldsymbol{x}}]_l\boldsymbol{x} \geq [b^E]_l$, that is nontrivial, i.e., $[\boldsymbol{A}^{E,\boldsymbol{x}}]_l \neq \boldsymbol{0}^T$. Indeed, if such a constraint existed, then for an $\boldsymbol{x}$ not satisfying this constraint we would have $\inf\{z : (\boldsymbol{x}, z) \in E\} = \infty$, which is a contradiction. On the other hand if there were a constraint with $[\boldsymbol{A}^{E,\boldsymbol{x}}]_l\boldsymbol{x} + [\boldsymbol{a}^{E,z}]_l z \geq [\boldsymbol{b}^E]_l$, with $[\boldsymbol{a}^{E,z}]_l < 0$ then, for all $(\boldsymbol{x}, z) \in E$ we would have $z \leq (-[\boldsymbol{A}^{E,\boldsymbol{x}}]_l\boldsymbol{x} + [b^E]_l)/[\boldsymbol{a}^{E,z}]_l$ and thus $\sup\{z : (\boldsymbol{x}, z) \in E\} < \infty$.

It is then easy to see that the function q given by (11) has as epigraph the set E. Therefore, it is the quotient.

To determine the remainder it is sufficient to find all the terms i of $p(\boldsymbol{x})$ for which $p(\boldsymbol{x}) > \tilde{p}(\boldsymbol{x})$, for an $\boldsymbol{x} \in \mathbb{R}^n$ that the term i attains the maximum in $p(\boldsymbol{x})$. The function $p(\boldsymbol{x}) - \tilde{p}(\boldsymbol{x})$ is linear on $P_{i,j}$ for all i, j. Therefore, since $p(\boldsymbol{x}) - \tilde{p}(\boldsymbol{x}) \geq 0$, and $\boldsymbol{x}_{i,j}$ is in the relative interior of $P_{i,j}$ we have either $p(\boldsymbol{x}) - \tilde{p}(\boldsymbol{x}) = 0$, for all $\boldsymbol{x} \in P_{i,j}$, or $p(\boldsymbol{x}_{i,j}) > \tilde{p}(\boldsymbol{x}_{i,j})$. Hence, the procedure described in Algorithm 1 produces the remainder.

F. *Proof of Proposition 5*

Observe that

$$p(\boldsymbol{x}_j) - d(\boldsymbol{x}_j) - q_t(\boldsymbol{x}_j) = f(\boldsymbol{x}_j) - q_t(\boldsymbol{x}_j) \geq 0,$$

where the last inequality is a consequence of the first constraint of (16). Furthermore,

$$e(t) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) - \sum_{j=1}^{N} \bigvee_{i=1}^{\tilde{m}_q} (\hat{\boldsymbol{a}}_{i,t}^T \boldsymbol{x}_j + \hat{b}_{i,t}), \tag{33}$$

where $\hat{\boldsymbol{a}}_{i,t}$ is the solution of (17) at the iteration t. Each term of the last sum can be written as:

$$\bigvee_{i=1}^{\tilde{m}_q} (\hat{\boldsymbol{a}}_{i,t}^T \boldsymbol{x}_j + \hat{b}_{i,t}) = (\hat{\boldsymbol{a}}_{i'_{j,t},t}^T \boldsymbol{x}_j + \hat{b}_{i'_{j,t},t}),$$

where $i'_{j,t}$ is the partition that j belongs after step t, i.e., $j \in I_{i'_{j,t},t}$ and $I_{\cdot,t}$ is the partition obtained at Step 6 in iteration t. Furthermore due to the optimization in (16),

$$\sum_{j=1}^{N} \hat{\boldsymbol{a}}_{i'_{j,t},t}^T \boldsymbol{x}_j + \hat{b}_{i'_{j,t},t} \leq \sum_{j=1}^{N} \hat{\boldsymbol{a}}_{i'_{j,t},t+1}^T \boldsymbol{x}_j + \hat{b}_{i'_{j,t},t+1}.$$

From Step 6,

$$\sum_{j=1}^{N} \hat{\boldsymbol{a}}_{i'_{j,t},t+1}^T \boldsymbol{x}_j + \hat{b}_{i'_{j,t},t+1} \leq \sum_{j=1}^{N} \hat{\boldsymbol{a}}_{i'_{j,t+1},t+1}^T \boldsymbol{x}_j + \hat{b}_{i'_{j,t+1},t+1}.$$

Therefore, $\sum_{j=1}^{N} \bigvee_{i=1}^{\tilde{m}_q} (\hat{\boldsymbol{a}}_{i,t}^T \boldsymbol{x}_j + \hat{b}_{i,t})$ is non-increasing, and thus, $e(t)$ is non-decreasing.

G. *Proof of Proposition 6*

We only prove (20), the proof of the other statements is similar. The quotients $Q(p_1, d)$ and $Q(p_2, d)$ satisfy

$$p_1(\boldsymbol{x}) \geq d(\boldsymbol{x}) + Q(p_1, d)(\boldsymbol{x}), \tag{34}$$
$$p_2(\boldsymbol{x}) \geq d(\boldsymbol{x}) + Q(p_2, d)(\boldsymbol{x}). \tag{35}$$

Thus,

$$p_1(\boldsymbol{x}) + p_2(\boldsymbol{x}) \geq d(\boldsymbol{x}) + [Q(p_1, d) + Q(p_2, d)(\boldsymbol{x}) + d(\boldsymbol{x})].$$

But $Q(p_1 + p_2, d)$ is the maximum tropical polynomial satisfying this inequality. Thus it is greater than or equal to the expression in the bracket, i.e.,

$$Q(p_1 + p_2, d)(\boldsymbol{x}) \geq Q(p_1, d)(\boldsymbol{x}) + Q(p_2, d)(\boldsymbol{x}) + d(\boldsymbol{x}).$$

References

[1] D. Maclagan and B. Sturmfels, "Introduction to tropical geometry," *Graduate Studies in Mathematics*, vol. 161, pp. 75–91, 2009.

[2] P. Maragos, V. Charisopoulos, and E. Theodosis, "Tropical geometry and machine learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 728–755, 2021.

[3] L. Zhang, G. Naitzat, and L.-H. Lim, "Tropical geometry of deep neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5824–5832.

[4] V. Charisopoulos and P. Maragos, "A tropical approach to neural networks with piecewise linear activations," *arXiv preprint arXiv:1805.08749*, 2018.

[5] G. Montúfar, Y. Ren, and L. Zhang, "Sharp bounds for the number of regions of maxout networks and vertices of Minkowski sums," *arXiv preprint arXiv:2104.08135*, 2021.

[6] M. Alfarra, A. Bibi, H. Hammoud, M. Gaafar, and B. Ghanem, "On the decision boundaries of neural networks: A tropical geometry perspective," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[7] C. Hertrich, A. Basu, M. Di Summa, and M. Skutella, "Towards lower bounds on the depth of ReLU neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3336–3348, 2021.

[8] Z. Li and C. Wang, "Achieving sharp upper bounds on the expressive power of neural networks via tropical polynomials," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[9] M. Trimmel, H. Petzka, and C. Sminchisescu, "TropEx: An algorithm for extracting linear terms in deep neural networks," in *International Conference on Learning Representations*, 2020.

[10] G. Smyrnis and P. Maragos, "Tropical polynomial division and neural networks," *arXiv preprint arXiv:1911.12922*, 2019.

[11] G. Smyrnis, P. Maragos, and G. Retsinas, "Maxpolynomial division with application to neural network simplification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4192–4196.

[12] G. Smyrnis and P. Maragos, "Multiclass neural network minimization via tropical Newton polytope approximation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9068–9077.

[13] P. Misiakos, G. Smyrnis, G. Retsinas, and P. Maragos, "Neural network approximation based on Hausdorff distance of tropical zonotopes," in *International Conference on Learning Representations*, 2021.

[14] K. Fotopoulos, P. Maragos, and P. Misiakos, "TropNNC: Structured neural network compression using tropical geometry," *arXiv preprint arXiv:2409.03945*, 2024.

[15] G. X. Ritter and P. Sussner, "An introduction to morphological neural networks," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 4. IEEE, 1996, pp. 709–717.

[16] G. X. Ritter and G. Urcid, "Lattice algebra approach to single-neuron computation," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 282–295, 2003.

[17] P. Sussner and E. L. Esmi, "Morphological perceptrons with competitive learning: Lattice-theoretical framework and constructive learning algorithm," *Information Sciences*, vol. 181, no. 10, pp. 1929–1950, 2011.

[18] V. Charisopoulos and P. Maragos, "Morphological perceptrons: geometry and training algorithms," in *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*. Springer, 2017, pp. 3–15.

[19] Y. Shen, X. Zhong, and F. Y. Shih, "Deep morphological neural networks," *arXiv preprint arXiv:1909.01532*, 2019.

[20] N. Dimitriadis and P. Maragos, "Advances in morphological neural networks: Training, pruning and enforcing shape constraints," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3825–3829.

[21] E. Theodosis and P. Maragos, "Analysis of the viterbi algorithm using tropical algebra and geometry," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.

[22] ——, "Tropical modeling of weighted transducer algorithms on graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8653–8657.

[23] P. Maragos and E. Theodosis, "Tropical geometry and piecewise-linear approximation of curves and surfaces on weighted lattices," *arXiv preprint arXiv:1912.03891*, 2019.

[24] ——, "Multivariate tropical regression and piecewise-linear surface fitting," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3822–3826.

[25] G. C. Calafiore, S. Gaubert, and C. Possieri, "Log-sum-exp neural networks and posynomial models for convex and log-log-convex data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 3, pp. 827–838, 2019.

[26] ——, "A universal approximation result for difference of log-sum-exp neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 12, pp. 5603–5612, 2020.

[27] K. H. Kim and F. W. Roush, "Factorization of polynomials in one variable over the tropical semiring," *arXiv preprint math/0501167*, 2005.

[28] N. Grigg and N. Manwaring, "An elementary proof of the fundamental theorem of tropical algebra," *arXiv preprint arXiv:0707.2591*, 2007.

[29] S. Gao and A. G. Lauder, "Decomposition of polytopes and polynomials," *Discrete & Computational Geometry*, vol. 26, no. 1, pp. 89–104, 2001.

[30] H. R. Tiwary, "On the hardness of computing intersection, union and Minkowski sum of polytopes," *Discrete & Computational Geometry*, vol. 40, no. 3, pp. 469–479, 2008.

[31] B. Lin and N. M. Tran, "Linear and rational factorization of tropical polynomials," *arXiv preprint arXiv:1707.03332*, 2017.

[32] R. A. Crowell, "The tropical division problem and the Minkowski factorization of generalized permutahedra," *arXiv preprint arXiv:1908.00241*, 2019.

[33] N. M. Tran and J. Wang, "Minimal representations of tropical rational signomials," *arXiv preprint arXiv:2205.05647*, 2022.

[34] M. Akian, S. Gaubert, V. Niţică, and I. Singer, "Best approximation in max-plus semimodules," *Linear Algebra and its Applications*, vol. 435, no. 12, pp. 3261–3296, 2011.

[35] A. Magnani and S. P. Boyd, "Convex piecewise-linear fitting," *Optimization and Engineering*, vol. 10, no. 1, pp. 1–17, 2009.

[36] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, "Synchronization and linearity: an algebra for discrete event systems," 1992.

[37] B. Heidergott, G. J. Olsder, and J. Van Der Woude, *Max plus at work*. USA: Princeton Univ. Press, 2014.

[38] P. Butkovič, *Max-linear systems: theory and algorithms*. Springer Science & Business Media, 2010.

[39] R. A. Cuninghame-Green, *Minimax algebra*. Springer Science & Business Media, 2012, vol. 166.

[40] P. Maragos, "Dynamical systems on weighted lattices: General theory," *Mathematics of Control, Signals, and Systems*, vol. 29, no. 4, pp. 1–49, 2017.

[41] D. Maclagan and B. Sturmfels, *Introduction to tropical geometry*. American Mathematical Society, 2021, vol. 161.

[42] M. Joswig, *Essentials of tropical combinatorics*. American Mathematical Society, 2021, vol. 219.

[43] I. Itenberg, G. Mikhalkin, and E. I. Shustin, *Tropical algebraic geometry*. Springer Science & Business Media, 2009, vol. 35.

[44] K. Fukuda, "Lecture: Polyhedral computation, spring 2016," *Dept. Math., Inst. Theor. Comput. Sci., ETH Zurich, Zurich, Switzerland, Lect. Notes*, 2020.

[45] V. Kaibel, "Extended formulations in combinatorial optimization," *arXiv preprint arXiv:1104.1023*, 2011.

[46] R. T. Rockafellar, "Convex analysis," in *Convex analysis*. Princeton university press, 2015.

[47] P. Maragos, "Morphological systems: slope transforms and max-min difference and differential equations," *Signal Processing*, vol. 38, no. 1, pp. 57–77, 1994.

[48] ——, "Slope transforms: theory and application to nonlinear signal processing," *IEEE Transactions on signal processing*, vol. 43, no. 4, pp. 864–877, 1995.

[49] H. J. Heijmans and P. Maragos, "Lattice calculus of the morphological slope transform," *Signal Processing*, vol. 59, no. 1, pp. 17–42, 1997.

[50] K. Fukuda *et al.*, "Frequently asked questions in polyhedral computation," *ETH, Zurich, Switzerland*, 2004.

[51] E. Balas, "Disjunctive programming: Properties of the convex hull of feasible points," *Discrete Applied Mathematics*, vol. 89, no. 1-3, pp. 3–44, 1998.

[52] J. Kim, L. Vandenberghe, and C.-K. K. Yang, "Convex piecewise-linear modeling method for circuit optimization via geometric programming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1823–1827, 2010.

[53] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[54] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *Proceedings of machine learning and systems*, vol. 2, pp. 129–146, 2020.

[55] N. Lee, T. Ajanthan, and P. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *International Conference on Learning Representations (ICLR)*, 2019.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[57] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. a data-driven control framework," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2017.

[58] U. Rosolia, X. Zhang, and F. Borrelli, "Simple policy evaluation for data-rich iterative tasks," in *2019 American control conference (ACC)*. IEEE, 2019, pp. 2855–2860.

[59] H. Xue, E. L. Zhu, J. M. Dolan, and F. Borrelli, "Learning model predictive control with error dynamics regression for autonomous racing," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 13 250–13 256.

[60] Y. Li, K. H. Johansson, J. Mårtensson, and D. P. Bertsekas, "Data-driven rollout for deterministic optimal control," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 2169–2176.

[61] B. Sturmfels, "On the Newton polytope of the resultant," *Journal of Algebraic Combinatorics*, vol. 3, no. 2, pp. 207–236, 1994.

[62] M. Joswig, "The Cayley trick for tropical hypersurfaces with a view toward ricardian economics," in *Homological and Computational Methods in Commutative Algebra*. Springer, 2017, pp. 107–128.

[63] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

Ioannis Kordonis oannis Kordonis received his Diploma and Ph.D. in Electrical and Computer Engineering from the National Technical University of Athens (NTUA), Greece, in 2009 and 2015, respectively. He held postdoctoral positions at the University of Southern California, CentraleSupélec, and NTUA. He is currently an Assistant Professor at the NTUA School of ECE.

His research interests include Game Theory, Stochastic Control Theory, and Machine Learning, with applications in Energy and Power Systems, Transportation Systems, and Bioengineering.

Petros Maragos received his Ph.D. degree from Georgia Tech, Atlanta, in 1985. Then, he joined the faculty of the Division of Applied Sciences at Harvard University, where he worked for 8 years as a professor of EE affiliated with the Harvard Robotics Lab. In 1993 he joined the faculty of the School of ECE at Georgia Tech, affiliated with its Center for Signal & Image Processing. During periods of 1996-98, he had a joint appointment as director of research at the Institute of Language and Speech Processing in Athens. Since 1999, he has been working as a professor at the NTUA, where he is currently the director of the Intelligent Robotics & Automation Lab. He has held visiting positions at MIT in 2012, at UPenn in 2016, and at USC in 2023. He is a co-founder and since 2023 the acting director of the Robotics Institute at Athena RC. He is also a co-founder and since 2025 the director of the newly launched Hellenic Robotics Center of Excellence. His research and teaching interests include computer vision, speech & language, machine learning, and robotics. He is the recipient of several awards including an NSF Presidential Young Investigator Award, Best Paper awards from IEEE journals and Computer Vision conferences, and the Technical Achievement award from EURASIP. For his research contributions he was elected a Fellow of IEEE in 1995 and a Fellow of EURASIP in 2010. He has served as IEEE Distinguished Lecturer for 2017-18. Since 2023 he is a Life Fellow of IEEE.