

Bachelor Thesis

**Tropical Geometry of
Deep Neural Networks**

David Leatham
August 18, 2020

Contents

| | | |
|----------|---|-----------|
| 0 | Introduction | 2 |
| 1 | Tropical Algebra | 4 |
| 2 | Neural networks | 12 |
| 3 | Tropical algebra of neural networks | 22 |
| 4 | Tropical hypersurfaces | 26 |
| 5 | Tropical geometry of neural networks | 31 |

0 Introduction

Tropical geometry is a modern branch of mathematics subject to a rise in research throughout the past 30 years. Neural networks have found application in many disciplines and have been used to algorithmically manage complex tasks like self driving cars, face and text recognition or automatically processing hand written checks at banks. Neural networks impact our lives on a daily basis.

In this thesis we describe a new representation of neural networks through tropical geometry, following the recent article "Tropical Geometry of Deep Neural Networks" by Liwen Zhang, Gregory Naitzat and Lek-Heng Lim.

The subject of neural networks was introduced by Warren McCulloch and Walter Pitts in 1943 when they created a computational model for neural networks [MP43]. In the following years one of the next big steps towards today's state of the art neural networks, was the research into the underlying concepts of backpropagation in 1960 by Kelley [Kel60] and 1961 by Bryson [Bry61]. Backpropagation is an algorithm describing a key concept of training neural networks. Starting 2010 to this day neural networks are experiencing a boost in popularity. This is due to hardware improvements especially in GPU's making backpropagation feasible [CMGS10], the growing amount of data and the improvement of knowledge on deep neural networks fuelled by articles and works, for example that of Andrew Yan-Tak Ng and Jeff Dean in 2012. They created a network that learned to recognize higher-level concepts, such as cats, only from watching unlabeled images [LMD⁺11]. Nowadays neural networks help solve challenging problems in computer science and software engineering and, in particular, they are key to application areas such as system identification and control, pattern and sequence recognition, social network filtering and e-mail spam filtering.

Tropical geometry on the other hand is a field of interest in mathematics, whose underlying analytical ideas have been around some years before 1990, for example by Victor Pavlovich Maslov [Mas85]. But only since then have there been basic efforts to consolidate the basic definitions as research increased and from 2000 onwards gained the attention it receives nowadays carried by successful applications in enumerative tropical algebraic geometry for instance by Grigory Mikhalkin in 2005 [Mik05]. Tropical geometry is the study of polynomials under tropical multiplication and addition, where tropical multiplication is usual addition and tropical addition of two elements returns the minimal element. It is used in mathematics itself [Kri14], finance [Kle09] and computer science. For the last topic this thesis is an example, as we describe a fundamental mathematical connection between tropical geometry and deep neural networks.

Concerning the structure of the thesis we start by introducing the reader to the necessary theories of tropical geometry in chapter 1 and the theory of neural networks in chapter 2, but also extend the exploration of the topics beyond the scope needed for the fundamental Theorems in this thesis to give a more

rounded introduction to these topics themselves when appropriate. Chapter 3 uses the knowledge of the first two chapters to demonstrate a fundamental link between neural networks and the difference between two tropical polynomials, called tropical rational functions, which is the core insight of the thesis. Then, to put this knowledge to use, we formulate statements about decision boundaries of neural networks in chapter 5 with tropical geometry, or more concrete, the link between neural networks and tropical rational functions. This needs fundamental knowledge of tropical hypersurfaces introduced in chapter 4.

This thesis follows the underlying paper "Tropical Geometry of Deep Neural Networks" by Liwen Zhang, Gregory Naitzat and Lek-Heng Lim very closely, in that all main statements may be reordered and rewritten, but come directly out of [MS15] and are cited accordingly.

1 Tropical Algebra

We begin by introducing tropical algebra. The remaining chapters will build upon chapter 1 without exception. The main goal is to establish an intuition and formal understanding of subsets of functions we call tropical and tropical rational functions if the image is one dimensional and tropical and tropical rational maps if not.

Our basic object of study is the semifield $(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$. As a set this corresponds to the real numbers \mathbb{R} , together with an extra element $-\infty$ which represents minus infinity. In this semifield both operations are defined on two elements $x, y \in \mathbb{R} \cup \{-\infty\}$. The first operation is called the **tropical sum** and returns the maximum of two elements, while the second operation is called **tropical product** and returns the usual sum:

$$x \oplus y := \max(x, y) \quad \text{and} \quad x \odot y := x + y.$$

To be in a position to understand the properties of $(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$, we define a semiring and a semifield formally.

Definition 1.1. [BP85] A **semiring** is a set \mathcal{R} equipped with two binary operations $+$ and \cdot , called addition and multiplication, such that for $a, b, c \in \mathcal{R}$

- $(\mathcal{R}, +)$ is a commutative monoid with identity element 0, meaning the following holds:
 - $(a + b) + c = a + (b + c)$
 - $0 + a = a + 0 = 0$
 - $a + b = b + a$
- (\mathcal{R}, \cdot) is a monoid with identity element 1, meaning:
 - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
 - $1 \cdot a = a \cdot 1 = a$
- Multiplication left and right distributes over addition:
 - $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
 - $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$
- Multiplication by 0 annihilates:
 - $0 \cdot a = a \cdot 0 = 0$

Proposition 1.2. The object $(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$ is a semiring. [MS15, p. 10]

Proof. The neutral element for the tropical sum is $-\infty$ since for $x \in \mathbb{R} \cup \{-\infty\}$, $x \oplus \infty = \max(x, -\infty) = x$ holds and with $x \odot 0 = x + 0 = x$ for $x \in \mathbb{R}$, 0 is the neutral element of tropical multiplication. Both tropical addition and multiplication are commutative. This is obvious for $x, y \in \mathbb{R}$, so take $y \in \mathbb{R} \cup \{-\infty\}$ and with the following all cases are shown:

$$\begin{aligned} -\infty \oplus y &= \max(-\infty, y) = y = \max(y, -\infty) = y \oplus \infty \\ \infty \odot y &= \infty + y = \infty = y + \infty = y \odot \infty. \end{aligned}$$

Tropical multiplication distributes over addition since if we take $x, y, z \in \mathbb{R}$ then

$$\begin{aligned} x \odot (y \oplus z) &= x + \max(y, z) = \max(x + y, x + z) = (x \odot y) \oplus (x \odot z) \\ (y \oplus z) \odot x &= \max(y, z) + x = \max(y + x, z + x) = (y \odot x) \oplus (z \odot x) \end{aligned}$$

holds. Multiplication by $-\infty$ annihilates $-\infty \odot x = -\infty \forall x \in \mathbb{R} \cup \{-\infty\}$ and therefore \mathbb{T} is a semiring. \square

Definition 1.3. [OGAJ20] A commutative semiring in which every nonzero element is multiplicatively invertible is called a **semifield**.

Tropical multiplication corresponds to usual addition which is commutative and invertible for all $x \in \mathbb{R}$. As shown in 1.2 the elements $-\infty$ represents zero. This shows the following proposition.

Proposition 1.4. The object $(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$ is a semifield.

Definition 1.5. We call the semifield $\mathbb{T} := (\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$ the **tropical semifield**.

In tropical geometry often infinity instead of minus infinity and min instead of max are used. This does not change any of the underlying theories of tropical algebra as the two semirings are tropically isomorph. This means

$$\begin{aligned} (\mathbb{R} \cup \{-\infty\}, \oplus := \max, \odot) &\rightarrow (\mathbb{R} \cup \{\infty\}, \oplus := \min, \odot), \\ x &\mapsto \begin{cases} -x & x \in \mathbb{R} \\ \infty & x = -\infty \end{cases} \end{aligned}$$

is an Isomorphism of semifields.

An essential feature of tropical arithmetics is that there is no subtraction. Take $a, b \in \mathbb{R} \cup \{-\infty\}$ with $a < b$ then the equation $a \oplus x = b$ has no solution x at all. [MS15, p. 11]

To get more familiar with the tropical arithmetic we will have a look at an example before we introduce tropical polynomials.

Remark 1.6. We will examine the tropical Pascal's triangle, whose rows are the coefficients appearing in a binomial expansion [MS15]. This means that the n -th layer of the triangle consists of $n + 1$ entries corresponding to p_0, \dots, p_n in:

$$(a \oplus b)^n = (p_0 \odot a^n) \oplus (p_1 \odot a^{n-1} \odot b) \oplus \dots \oplus (p_{n-1} \odot a \odot b^{n-1}) \oplus (p_n \odot b^n).$$

for $a, b \in \mathbb{T}$. Let us calculate the entries for the fourth row and set $a, b \in \mathbb{T}$ again.

$$\begin{aligned}
(a \oplus b)^3 &= 3 \max(a, b) \\
&= \max(3a, 3b) = (a^3 + b^3) \\
&= \max(0 \odot a^3, 0 \odot a^2 \odot b, 0 \odot a \odot b^2, 0 \odot b^3) \\
&= (0 \odot a^3) \oplus (0 \odot a^2) \odot (b \oplus 0 \odot a) \odot (b^2 \oplus 0 \odot b^3).
\end{aligned}$$

You may say the Pascal's coefficients are four zeros. And actually the same applies to all cases.

$$\begin{aligned}
(a \oplus b)^n &= a^n \oplus b^n \\
&= (0 \odot a^n) \oplus (0 \odot a^{n-1} \odot b) \oplus \dots \oplus (0 \odot a \odot b^{n-1}) \oplus (0 \odot b^n).
\end{aligned}$$

And therefore the tropical Pascal's triangle, whose rows are the coefficients appearing in a binomial expansion take a simple shape. All its coefficients are zero.

$$\begin{array}{cccccccc}
n = 0 & & & & & & & 0 \\
n = 1 & & & & & 0 & & 0 \\
n = 2 & & & & 0 & & 0 & 0 \\
n = 3 & & & 0 & & 0 & & 0 & 0 \\
n = 4 & & 0 & & 0 & & 0 & & 0 & 0 \\
n = 5 & & 0 & & 0 & & 0 & & 0 & 0 & 0 \\
n = 6 & 0 & & 0 & & 0 & & 0 & & 0 & 0 & 0.
\end{array}$$

Moving on, first, so that we can introduce multidimensional tropical polynomials properly, a notion of monomials is needed.

Definition 1.7. [ZNL18, p. 2] A **tropical monomial** in d variables x_1, \dots, x_d is a function $\mathbb{T}^d \rightarrow \mathbb{T}$ of the form

$$c \odot x_1^{a_1} \odot x_2^{a_2} \odot \dots \odot x_{d-1}^{a_{d-1}} \odot x_d^{a_d}$$

where $c \in \mathbb{R} \cup \{-\infty\}$ and $a_1, \dots, a_d \in \mathbb{N}$. As a convenient shorthand notation, we will also write a tropical monomial in multiindex notation as cx^α where $\alpha = (a_1, \dots, a_d) \in \mathbb{N}_d$ and $x = (x_1, \dots, x_d)$. Note that $x^\alpha = 0 \odot x^\alpha$.

Definition 1.8. [ZNL18, p. 2] A **tropical polynomial** $f : \mathbb{T}^d \rightarrow \mathbb{T}$ is a finite tropical sum of tropical monomials

$$f(x) = c_1 x^{\alpha_1} \oplus \dots \oplus c_r x^{\alpha_r}$$

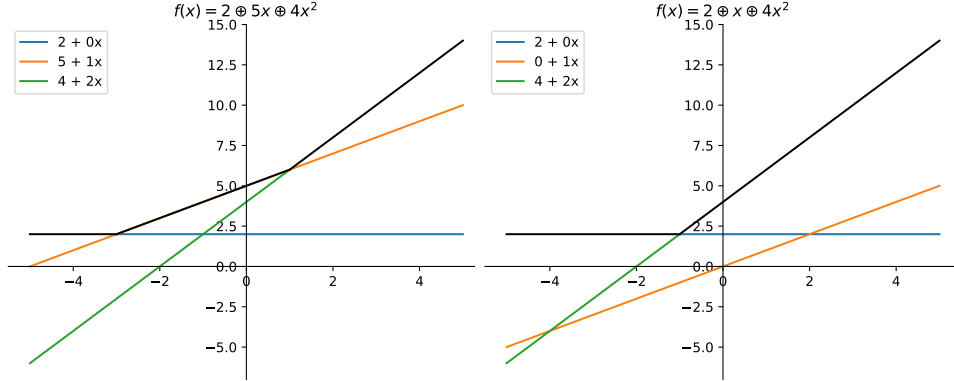
where $\alpha_i = (\alpha_{i1}, \dots, \alpha_{id}) \in \mathbb{N}^d$ and $c_i \in \mathbb{T}$, $i = 1, \dots, r$. We will assume that a monomial of a given multiindex appears at most once in the sum, i.e. $\alpha_i \neq \alpha_j$ for any $i \neq j$.

The condition $\alpha_i \neq \alpha_j$ with constant c_k 's for $k = 1, \dots, r$ does not restrict the tropical polynomial, but only standardises the form of representation since for $\alpha_i = \alpha_j$ the two monomials in f can be combined to one monomial without changing the function as follows

$$\begin{aligned} c_j x^{\alpha_j} \oplus c_j x^{\alpha_j} &= \max\{c_j + \alpha_j x, c_i + \alpha_i c_i\} \\ &= \begin{cases} c_j x^{\alpha_j} & \text{for } c_j \geq c_i \\ c_i x^{\alpha_i} & \text{for } c_j < c_i. \end{cases} \end{aligned}$$

Remark 1.9. Now we examine tropical polynomials in one variable. Monomials in one variable are of the form $c \odot x^a : \mathbb{T} \rightarrow \mathbb{T}$ where $c \in \mathbb{T}$ and $a \in \mathbb{N}$, which means a tropical polynomial in one variable is of the form $f(x) = c_1 x^{\alpha_1} \oplus \dots \oplus c_r x^{\alpha_r}$ which $c_i \in \mathbb{T}, \alpha_i \in \mathbb{N}$ for $i = 1, \dots, r$.

For a quadratic tropical polynomial $f(x) = a \oplus bx \oplus cx^2$ with $a, b, c \in \mathbb{T}$ linearity breaks at two points $b - c$ and $a - b$ if the condition $b - c > a$ holds and otherwise only breaks at one point $\frac{a-c}{2}$. Visualising two quadratic tropical functions gives a good intuition of the piecewise-linearity. The black coloured parts of Figures 1a and 1b indicate the graph of the tropical polynomial.



(a) Take $f(x) = 2 \oplus 5x \oplus 4x^2$ a quadratic polynomial. The graph equals $\max(2, 5 + x, 4 + 2x)$. Until $5 - 2$, 2 dominates, from there $5 + x$ dominates to the point $4 + 2x$.

(b) Take $f(x) = 2 \oplus x \oplus 4x^2$, we have changed the scalar in the second part, then this second part plays no part in the tropical polynomial.

Figure 1: Quadratic tropical polynomials.

We can see the degree of a tropical polynomial, defined the same as a degree of a usual polynomial, gives an upper bound for the number of non-linear edges of the tropical polynomial, but not the exact value. With higher degree polynomials more non linear edges are possible as figure 2 illustrates.

Definition 1.10. Let S be a set in a real vector space. Then S is **convex** if for $s_1, s_2 \in S$:

$$\lambda s_1 + (1 - \lambda) s_2 \in S$$

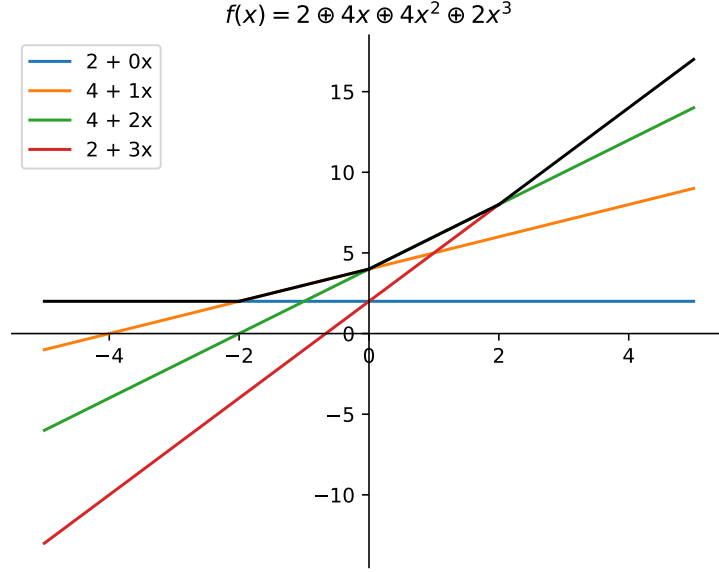


Figure 2: Take $f(x) = 2 \oplus x \oplus 4x^2 \oplus 4x^3$. For a degree three tropical polynomial in one variable this polynomial has reached the maximum number of non linear edges.

holds for all $\lambda \in [0, 1] \subset \mathbb{R}$.

Definition 1.11. Let X be a convex set in a real vector space and let $f : X \rightarrow \mathbb{R}$ be a function, then f is called **convex** if for $x_1, x_2 \in X$ and $\lambda \in [0, 1] \subset \mathbb{R}$, $f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$ holds.

Definition 1.12. A function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ with $n, m \in \mathbb{N}$ is **piecewise linear** if there exist closed sets $(P_i)_{i \in \mathbb{N}} \subset \mathbb{R}^m$ with $f : P_i \rightarrow \mathbb{R}^n$ linear and $\cup_i P_i = \mathbb{R}^m$.

In multiple variables we can characterise tropical polynomials as functions from $\mathbb{R}^n \rightarrow \mathbb{R}$ that satisfy the following three properties.

Lemma 1.13. Let f be a tropical polynomial

$$f(x) = c_1 x^{\alpha_1} \oplus \dots \oplus c_r x^{\alpha_r}$$

as in definition 1.8, but we restrict $c_1, \dots, c_r \in \mathbb{R}$ to be real and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ with $n \in \mathbb{N}$. Then f has three important properties:

- (1) f is continuous.
- (2) f is piecewise-linear, where the number of pieces is finite.
- (3) f is convex, i.e. $p(\frac{x+y}{2}) \leq \frac{1}{2}(p(x) + p(y)) \forall x, y \in \mathbb{R}$

Proof. The minimum of continuous functions is still continuous which proves (1). Every monomial $c_i x^{\alpha_i} = c_i + x_1 \alpha_{i1} + \dots + x_n \alpha_{in}$ is per definition linear. Because of (1), linearity can only be broken where $c_i x^{\alpha_i} = c_j x^{\alpha_j}$ for $i \leq j$ and $i, j = 1, \dots, r$. A piece is a single maximal region of $c_i x^{\alpha_i}$ on which linearity is not broken. If we introduce $c_i x^{\alpha_i}$ one after another, then in the i -th step not more than i^2 new pieces can be created, so there can only be $\sum_{i=1}^r i^2$ or less pieces which proves (2). For (3) let $x, y \in \mathbb{R}^n$, then:

$$\begin{aligned}
& f(\lambda x + (1 - \lambda)y) \\
&= \bigoplus_{i=1}^r c_i (\lambda x + (1 - \lambda)y)^{\alpha_i} \\
&= \max_{i=1}^r \{c_i + \sum_{j=1}^n \alpha_{ij} (\lambda x_j + (1 - \lambda)y_j)\} \\
&= \max_{i=1}^r \{\lambda(c_i + \sum_{j=1}^n \alpha_{ij} x_j) + (1 - \lambda)(c_i + \sum_{j=1}^n \alpha_{ij} y_j)\} \\
&\leq \max_{i=1}^r \{\lambda(c_i + \sum_{j=1}^n \alpha_{ij} x_j)\} + \max_{i=1}^r \{(1 - \lambda)(c_i + \sum_{j=1}^n \alpha_{ij} y_j)\} \\
&= \lambda \max_{i=1}^r \{c_i + \sum_{j=1}^n \alpha_{ij} x_j\} + (1 - \lambda) \max_{i=1}^r \{c_i + \sum_{j=1}^n \alpha_{ij} y_j\} \\
&= \lambda f(x) + (1 - \lambda)f(y)
\end{aligned}$$

which ends the prove. \square

The same could be argued for our tropical function f , if we let $c_i \in \mathbb{T}$ and $f : \mathbb{T}^n \rightarrow \mathbb{T}$, but the underlying theories, for example for linearity, change for semifields [OGAJ20].

Proposition 1.14. Every function $\mathbb{R}^n \rightarrow \mathbb{R}$ which satisfies the three properties (1), (2) and (3) from lemma 1.13 has a representation as the minimum of a finite set of linear functions. Thus, the tropical polynomials in n variables x_1, \dots, x_n represent the class of piecewise-linear convex functions on \mathbb{R}^n with integer coefficients.

Proof. This follows directly from lemma 1.13. \square

Now we are ready to introduce tropical rational functions and the semifields $\mathbb{T}[X_1, \dots, X_d]$, $\mathbb{T}(X_1, \dots, X_d)$. These are important to understand the core section (section 3), in particular the actual connection build between neural networks and tropical geometry in this thesis.

Definition 1.15. [ZNL18, p. 3] A **tropical rational function** is a standard difference, or, equivalently, a tropical quotient of two tropical polynomials $f(x)$ and $g(x)$:

$$(f - g)(x) = f(x) - g(x) = f(x) \oslash g(x) = (f \oslash g)(x)$$

Proposition 1.16. $\mathbb{T}[X_1, \dots, X_d] := \{f : \mathbb{T}^d \rightarrow \mathbb{T}; f \text{ is tropical polynomial}\}$ and $\mathbb{T}(X_1, \dots, X_d) := \{f : \mathbb{T}^d \rightarrow \mathbb{T}; f \text{ is tropical rational function}\}$ are both semirings. [ZNL18, p. 3]

Proof. We begin the proof by showing, that for two tropical polynomials $a, b \in \mathbb{T}[X_1, \dots, X_r]$ with $a(x) = \bigoplus_{i=0}^{\alpha} z_i x^{\zeta_i}$ and $b(x) = \bigoplus_{i=0}^{\beta} o_i x^{\omega_i}$ their tropical multiplication is a tropical polynomial $(a \odot b)(x) \in \mathbb{T}[X_1, \dots, X_r]$.

$$\begin{aligned}
(a \odot b)(x) &= a(x) + b(x) \\
&= \left(\bigoplus_{i=0}^{\alpha} z_i x^{\zeta_i} \right) + \left(\bigoplus_{i=0}^{\beta} o_i x^{\omega_i} \right) \\
&= \max_{i=0}^{\alpha} \{z_i + \sum_{k=1}^r \zeta_{ik} x_k\} + \max_{j=0}^{\beta} \{o_j + \sum_{k=1}^r \omega_{jk} x_k\} \\
&= \max_{\substack{i=1, \dots, \alpha \\ j=1, \dots, \beta}} \{z_i + \sum_{k=1}^r \zeta_{ik} x_k + o_j + \sum_{k=1}^r \omega_{jk} x_k\} \\
&= \max_{\substack{i=1, \dots, \alpha \\ j=1, \dots, \beta}} \{(z_i + o_j + \sum_{k=1}^r (\zeta_{ik} + \omega_{jk}) x_k)\} \\
&= \bigoplus_{\substack{i=1, \dots, \alpha \\ j=1, \dots, \beta}} (z_i \odot o_j) \odot x^{\zeta_i + \omega_j} \in \mathbb{T}[X_1, \dots, X_r]
\end{aligned}$$

That $(a \oplus b) \in \mathbb{T}[X_1, \dots, X_r]$ is immediate. We have already shown that \mathbb{T} is a tropical semifield. Therefore, except for the neural elements, all other axioms hold automatically pointwise.

The neutral element for the tropical sum is $-\infty = -\infty \odot x = x \odot -\infty \in \mathbb{T}$. Both $0, -\infty \in \mathbb{T}$ are also a tropical and tropical rational function. Since for $f(x) \in \mathbb{T}(X_1, \dots, X_r)$ or $f(x) \in \mathbb{T}[X_1, \dots, X_r]$, the following hold

$$\begin{aligned}
f(x) \oplus \infty &= \max(f(x), -\infty) = f(x) \\
f(x) \odot 0 &= f(x) + 0 = f(x).
\end{aligned}$$

Therefore 0 is the neutral element of tropical multiplication for tropical and tropical rations functions and $-\infty$ is the neural element of addition. We have shown $\mathbb{T}[X_1, \dots, X_r]$ is a semifield.

Let $g, f, h \in \mathbb{T}(X_1, \dots, X_d)$ with

$$f(x) = f_1(x) \odot f_2(x) g(x) = g_1(x) \odot g_2(x) h(x) = h_1(x) \odot h_2(x)$$

The tropical sum of tropical rational functions is a tropical rational function:

$$\begin{aligned}
(f \oplus g)(x) &= f(x) \oplus g(x) \\
&= (f_1(x) \odot f_2(x)) \oplus (g_1(x) \odot g_2(x)) \\
&= \min\{f_1(x) - f_2(x), g_1(x) - g_2(x)\} \\
&= \min\{f_1(x) + g_2(x), g_1(x) + f_2(x)\} - f_2(x) - g_2(x) \\
&= (f_1(x) + g_2(x) \oplus g_1(x) + f_2(x)) \\
&\quad \odot (f_2(x) + g_2(x)) \in \mathbb{T}(X_1, \dots, X_r).
\end{aligned}$$

since addition as tropical addition of tropical polynomials is a tropical polynomial. The tropical product of tropical rational functions is a tropical rational function:

$$\begin{aligned}
(f \odot g)(x) &= f(x) \odot g(x) \\
&= (f_1(x) \odot f_2(x)) \odot (g_1(x) \odot g_2(x)) \\
&= f_1(x) - f_2(x) + g_1(x) - g_2(x) \\
&= (f_1(x) + g_1(x)) - (f_2(x) + g_2(x)) \in \mathbb{T}(X_1, \dots, X_r)
\end{aligned}$$

This concludes the proof. \square

We regard a tropical polynomial $f = f \odot 0$ as a special case of a tropical rational function and thus $\mathbb{T}[X_1, \dots, X_r] \subseteq \mathbb{T}(X_1, \dots, X_r)$. [ZNL18, p. 3]

Remark 1.17. A d-variate tropical polynomial $f(x)$ defines a function $f : \mathbb{T}^d \rightarrow \mathbb{T}$ that is a convex function in the usual sense as taking max and \sum of convex functions preserve convexity [BBV04].

Definition 1.18. $R : \mathbb{R}^d \rightarrow \mathbb{R}^p, x = (x_1, \dots, x_d) \mapsto (f_1(x), \dots, f_p(x))$, is called a **tropical polynomial map** if each $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ is a tropical polynomial, $i = 1, \dots, p$, and a **tropical rational map** if f_1, \dots, f_p are tropical rational functions. We will denote the set of tropical polynomial maps by $Pol(d, p)$ and the set of tropical rational maps by $Rat(d, p)$. So $Pol(d, 1) = \mathbb{T}[X_1, \dots, X_d]$ and $Rat(d, 1) = \mathbb{T}(X_1, \dots, X_d)$ [ZNL18, p. 3].

Next we introduce deep neural networks.

2 Neural networks

Neural networks viewed as a topic, make for a very compelling field of interest in their own right. Historically the term "Neural Network" was introduced in attempts to describe the functionality of biological processes, in particular the nervous system and the brain, in a mathematical sense [MP43,WH60,RHW86]. Simplified the nervous system is a net of neurons, each having a soma and an axon. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse. This, except for the fact and the time of its occurrence, is determined by the neuron, not by the excitation. From the point of excitation the impulse is propagated to all parts of the neuron [MP43]. Through synapses the axons are connected to further soma through which the impulse is passed on to further neurons. Impulses passing through the nervous system partly consist of electrical impulses and chemical reactions [Pal56]. A collection of partially connected neurons, capable of carrying impulses, is called a biological neural network.

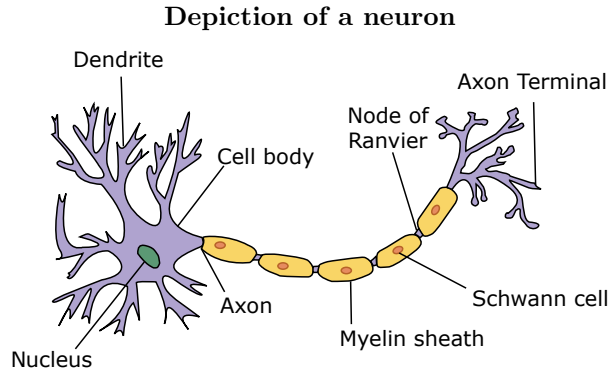


Figure 3: A representation of a neuron. The axon terminal attach to dendrite. This way impulses pass from neuron to neuron.

In reference to a biological neural network, an abstract neuronal network, in the following simply referenced as neuronal networks, are defined. As a biological neuron and especially a biological neural network are far more complex than this introduction may make it seem, biological realism would impose entirely unnecessary constraint. Boiling a biological neuronal network down to its quintessential features results in a weighted directed graph, where edges and vertices are weighted. Typically a weighted graph only has its edges weighted. To fit our model better we introduce them also with weighted vertices.

Definition 2.1. A graph is a pair $G = (V, E)$, where V is a set whose elements are called vertices, and $E \subset \{x, y | (x, y) \in v^2\}$ is a set of two-sets of vertices, whose elements are called edges.

Definition 2.2. A directed graph is a pair $G = (V, E)$, where V is a set whose

elements are called vertices, and $E \subset \{(x, y) | (x, y) \in v^2\}$ is a set of edges which are ordered pairs of distinct vertices.

Definition 2.3. A weighted graph $G = (V, E)$ in our case is attributed by two functions $\psi : V \rightarrow \mathbb{K}$ and $\omega : E \rightarrow \mathbb{K}$ that assign a weight $\psi(v)$ and $\omega(e)$ to each edge $e \in E$ and weight $v \in V$, with \mathbb{K} being a field.

Definition 2.4. Let $G = (V, E)$ be a Graph. We set $n(G) = |V|$ to be the cardinality of vertices and $m(G) = |E|$ the cardinality of edges.

Graphs can represent a multitude of relations between objects. Like road maps of roads connecting cities. Or describe objects themselves like molecules.

Remark 2.5. For instance the following weighted directed Graph could depict a road map with cities as nodes connected by roads that are weighted with the distance between cities and we want to find the minimal distance from any city to city A.

Min distance weighted graph

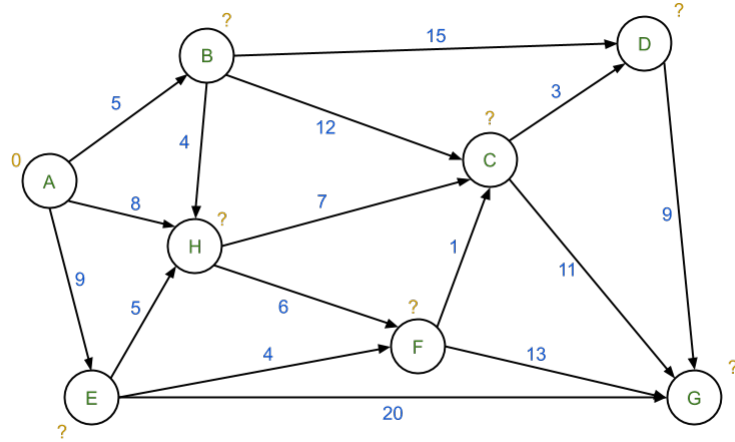


Figure 4: Min distance weighted graph.

In a graph where we have a way to reach each node from the starting point, as our graph has this property, Dijkstra's algorithm gives us, for small graphs, an easy algorithm to compute the shortest path from node A to any other node by hand. At any point in the algorithm simply consider all reachable nodes from nodes that have their shortest path already computed by adding up the shortest paths length to the previous node and the path length of the path between the nodes. This way every reachable node by nodes with there shortest path already computed get at least one or multiple lengths of paths assigned to them. Pick the shortest path under all of these and repeat until all vertices have a path

assigned. If there are multiple shortest paths, pick one of them.
 We will prove that this gives us a correct result through induction.

Proof. For each node v , $dist(v)$ is the shortest distance from source to v when travelling via visited nodes only, or infinity if no such path exists. (Note: we do not assume $dist(v)$ is the actual shortest distance for unvisited nodes.)

The base case is when there is just one visited node, namely the initial node source, in which case the hypothesis is trivial.

Otherwise assume the hypothesis for $n - 1$ visited nodes, in which case we choose an edge vu where u has the least $dist(u)$ of any unvisited nodes and the edge vu is such that $dist(u) = dist(v) + length(v, u)$. $dist(u)$ is considered to be the shortest distance from source to u because if there were a shorter path, and if w was the first unvisited node on that path then by the original hypothesis $dist(w) > dist(u)$ which creates a contradiction. Similarly if there were a shorter path to u without using unvisited nodes, and if the last but one node on that path were w , then we would have had $dist(u) = dist(w) + length(w, u)$, also a contradiction.

After processing u it will still be true that for each unvisited node w , $dist(w)$ will be the shortest distance from source to w using visited nodes only, because if there were a shorter path that doesn't go by u we would have found it previously, and if there were a shorter path using u we would have updated it when processing u .

After all nodes are visited, the shortest path from source to any node v consists only of visited nodes, therefore $dist(v)$ is the shortest distance. \square

Now that we understand Dijkstra's algorithm, to complete the remark, we will compute the shortest distance to vertex C , by terminating Dijkstra's algorithm as soon as we have computed the shortest path to C .

From A the shortest distance to adjacent vertices is the distance to vertex B with length 5. We set the shortest distance to B to 5 and repeat. This time we have to also consider the distances to adjacent vertices to B with the added shortest distance to B . So in this iteration paths of length 20 to D , 17 to C and 9 to H with is closely beaten by 8 to H from A are to be considered. We mark 8 to be the shortest distance to H . We describe one more step in detail and let the reader confirm if the calculated value for C is correct. We have now computed the minimal distance to nodes A , B and H . Reachable nodes at this point are D , C , F and E with path lengths of 20, 17, 15, 14 and 9. 9 is the shortest, the shortest way to node E is of length 9. Next is node F and then C with its length being 14.

Graph theory is a big field, but we are interested in modelling neural networks. In particular in those neural networks that have specific input layers and output layers. Our goal at this point is to get an insight of how to construct a neural network and define weights, so that our output stands in a predefined relation to our input. In particular one of the most important neural networks is the L-layer feedforward neural network. We will define and motivate L-layer

feedforward neural networks using graphs at first and then abstract again to only their necessary features.

Definition 2.6. An L -layer feedforward neural network in graph form (G, σ) is a weighted graph $G = (V, E)$ with an activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$. σ needs to be piecewise differentiable. The graph consists of L sets $V^{(j)}$, $j = 1, \dots, L$ of vertices ($V = \cup_{j=1}^L V^{(j)}$) with $L-1$ corresponding sets of edges $E^{(i)} \subset \{(x, y) | x \in V^{(i)}; y \in V^{(i+1)}\}$, $i = 1, \dots, L-1$ which connect two consecutive layers.

The Graph of a 0-layer feedforward neural network is an empty graph and of a 1-layer feedforward neural network is a set of vertices without connecting edges. σ will be applied in between propagating nodes in forward propagation 2.10.

Remark 2.7. The first layer of an L -layer feedforward neural network is called the input layer and the last (L -th) layer is called the output layer. All layers in between collectively are called hidden layers.

Remark 2.8. A fully connected feedforward neural network (G, σ) with $G = (V, E)$ is one where E is the largest out of every possible sets of E . This will give the unique solution, where $V^{(j)}$ and $V^{(j+1)}$ for each $j = 1, \dots, L-1$ will be fully connected.

We are at an interesting point of this chapter. Inspired by biological neural networks, which are the core building blocks of animal brains, we have abstracted some of the essential features, given an introduction to graphs and used these as a mathematical buildingblock for neural networks. With the introduction of feedforward neural networks we have cast a sensible form of neural network which is able to store and also learn complex relations between input data and output data through forward and backward propagation.

Forward propagation describes the process by which from an input vector the output vector is calculated. First we need to introduce the bias and weight vector $W^{(j)}$.

Remark 2.9. Let (G, σ) be a L -layer feedforward neural network where we fixate the elements from the sets $V^{(j)}$, by naming them b_{j1} to $b_{j|V^{(j)}|}$, so that we can establish a correlating vector $b^{(j)} = (b_{j1}, \dots, b_{j|V^{(j)}|})^t$, we call this vector the bias of layer j , which corresponds to the weights of vertices in $V^{(j)}$ for $j = 2, \dots, L$ and a matrix $W^{(j)}$ with the entries $W_{nm}^{(j)} = \omega((b_{j-1n}, b_{jnm}))$ which correspond to the weights of edges connecting layer $j-1$ and j where j is still in range $j = 2, \dots, L$.


We will label feedforward neural networks this way until we define feedforward neural networks as functions and are now in a position to define forward propagation.

Definition 2.10. (Forward Propagation) Let (G, σ) be as in 2.9. A Forward Propagation of our L -layer feedforward neural network (G, σ) and an input vector x , corresponds to the value of $a^{(L)}$, where $z^{(j)} = W^{(j)}a^{(j-1)} + b^{(j)}$, $a^{(j)} = \sigma(z^{(j)})$, for $j = 2, \dots, L$ and $a^{(1)} = x$.

Often binary inputs, as for example black and white pictures, are fed into neural networks and binary outputs, like dogs or cats are expected. For the sake of proving that the feed-forward neural networks we have defined are able to be set up, so that for any specific binary input any binary specific output can be obtained. We will view the infrastructure of the weights, bias and the function σ together as a circuit.


Remark 2.11. Binary circuits are defined by basic logical gates like AND, OR, and NOT gates and a lot more, that manipulate binary input as shown in figure 5. A set of gates is called a Universal Logic Gates Set if with those gates any other logic or boolean function can be created. After [Qui55] AND, OR and NOT define such a set. Meaning if we can reproduce these three gates in form of feedforward neural networks, any boolean logic can be created by these. For that we set $\omega = \mathbf{1}$, the weights of edges to one and

YES




| INPUT | OUTPUT |
|-------|--------|
| A | A |
| 0 | 0 |
| 1 | 1 |

NOT




| INPUT | OUTPUT |
|-------|--------|
| A | NOT A |
| 0 | 1 |
| 1 | 0 |

AND




| INPUT | | | OUTPUT |
|-------|---|--|--------|
| A | B | | |
| 0 | 0 | | 0 |
| 1 | 0 | | 0 |
| 0 | 1 | | 0 |
| 1 | 1 | | 1 |

OR




| INPUT | | | OUTPUT |
|-------|---|--|--------|
| A | B | | |
| 0 | 0 | | 0 |
| 1 | 0 | | 1 |
| 0 | 1 | | 1 |
| 1 | 1 | | 1 |

XOR




| INPUT | | | OUTPUT |
|-------|---|--|--------|
| A | B | | |
| 0 | 0 | | 0 |
| 1 | 0 | | 1 |
| 0 | 1 | | 1 |
| 1 | 1 | | 0 |

NAND




| INPUT | | | OUTPUT |
|-------|---|--|--------|
| A | B | | |
| 0 | 0 | | 1 |
| 1 | 0 | | 1 |
| 0 | 1 | | 1 |
| 1 | 1 | | 0 |

NOR



| INPUT | | | OUTPUT |
|-------|---|--|--------|
| A | B | | |
| 0 | 0 | | 1 |
| 1 | 0 | | 0 |
| 0 | 1 | | 0 |
| 1 | 1 | | 0 |

XNOR



| INPUT | | | OUTPUT |
|-------|---|--|--------|
| A | B | | |
| 0 | 0 | | 1 |
| 1 | 0 | | 0 |
| 0 | 1 | | 0 |
| 1 | 1 | | 1 |

Figure 5: Boolean Gates Truth Tables

$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0. \end{cases}$$

Usually the output of a neural network is non binary, but is trained so that the higher the output value is the higher the probability of the specific outcome and the lower the output the lower the probability, with zero being neutral. To skip this interpretation σ is defined to immediately produce a binary output. Now we will define three different 2-layer feedforward neural nets with this sigma.

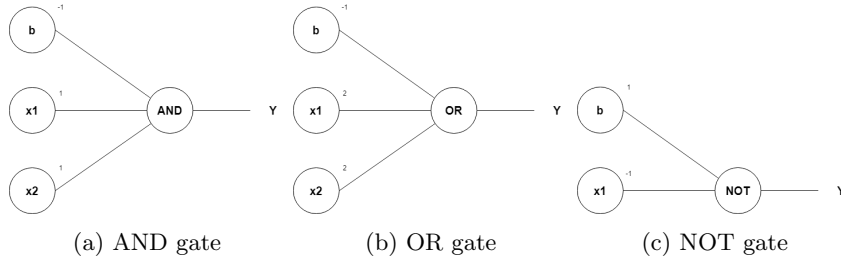


Figure 6: Neural Logical Gates

The first, motivated by figure 6 (a), defines our AND gate. It is defined with the graph $G = (E, V)$, $V = \{b_{11}, b_{12}, b_{21}, b_{31}\}$, $E = \{(b_{11}, b_{21}), (b_{12}, b_{21}), (b_{21}, b_{31})\}$

a fully connected neural network with weights

$$\psi(b_{11}) = \psi(b_{12}) = \psi(b_{31}) = 0, \psi(b_{21}) = -1 \text{ and } W^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, W^{(2)} = (1).$$

Lets compare the the boolean output with the truth table. Inputs $\{(x, y) | x, y \in \{0, 1\}\}$ with

$$z^{(1)} = (1, 1) \begin{pmatrix} x \\ y \end{pmatrix} - 1 \rightarrow z^{(2)} = (1)(\sigma(x + y - 1)) + 0 = \sigma(x + y - 1)$$

$$\begin{aligned} \text{for } (0, 0) \text{ yield } z^{(2)} &= \sigma(-1) = 0 \\ \text{for } (1, 0), (0, 1) \text{ yield } z^{(2)} &= \sigma(0) = 0 \\ \text{and for } (1, 1) \text{ yield } z^{(2)} &= \sigma(1) = 1. \end{aligned}$$

The same graph with weights

$$\psi(b_{11}) = \psi(b_{12}) = \psi(b_{31}) = 0, \psi(b_{21}) = -1 \text{ and } W^{(1)} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, W^{(2)} = (1)$$

makes an OR gate:

$$z^{(1)} = (2, 2) \begin{pmatrix} x \\ y \end{pmatrix} - 1 \rightarrow z^{(2)} = (1)(\sigma(2x + 2y - 1)) + 0 = \sigma(2x + 2y - 1)$$

$$\begin{aligned} \text{for } (0, 0) \text{ yield } z^{(2)} &= \sigma(-1) = 0 \\ \text{for } (1, 0), (0, 1) \text{ yield } z^{(2)} &= \sigma(2) = 1 \\ \text{and for } (1, 1) \text{ yield } z^{(2)} &= \sigma(4) = 1. \end{aligned}$$

And last but not least the graph of the NOT gate is different in that it only has one input. Therefore the graph also changes to $G = (E, V)$, $V = \{b_{11}, b_{21}, b_{31}\}$, $E = \{(b_{11}, b_{21}), (b_{21}, b_{31})\}$ also a fully connected 2 feedforward neural network. With weights

$$\psi(b_{11}) = \psi(b_{31}) = 0, \psi(b_{21}) = 1 \text{ and } W^{(1)} = (-1), W^{(2)} = (1)$$

that define a NOT gate:

$$\begin{aligned} \text{for } (0) \text{ yield } z^{(2)} &= z^{(1)} = 1 \\ \text{and for } (1) \text{ yield } z^{(2)} &= z^{(1)} = 0. \end{aligned}$$

We have proven that any binary input can create any binary output. This also means that by scaling the input and output weights it is possible from any specific input to create any specific output.

The last abstraction level of feedforward neural networks is to define them as a function, which enables us to compare feedforward neural networks to tropical polynomials in Chapter 4.

Definition 2.12. [ZNL18] Viewed abstractly, an L -layer feedforward neural network is a map $\nu : \mathbb{R}^d \rightarrow \mathbb{R}^p$ given by a composition of functions

$$\nu = \sigma^{(L)} \circ \rho^{(L)} \circ \sigma^{(L-1)} \circ \rho^{(L-1)} \circ \dots \circ \sigma^{(1)} \circ \rho^{(1)}$$

The preactivation functions $\rho^{(1)}, \dots, \rho^{(L)}$ are affine transformations to be determined and the activation functions $\sigma^{(1)}, \dots, \sigma^{(L)}$ are chosen and fixed in advanced.

We denote the width, i.e., the number of nodes, of the l th layer by $n_l, l = 1, \dots, L-1$. We set $n_0 := d$ and $n_L := p$, respectively the dimensions of the input and output of the network. The output from the l th layer will be denoted by

$$\nu = \sigma^{(l)} \circ \rho^{(l)} \circ \sigma^{(l-1)} \circ \rho^{(l-1)} \circ \dots \circ \sigma^{(1)} \circ \rho^{(1)}$$

i.e., it is a map $\nu^{(l)} : \mathbb{R}^d \rightarrow \mathbb{R}^{n_l}$. For convenience, we assume $\nu^{(0)}(x) := x$.

The affine function $\nu^{(l)} : \mathbb{R}^{n_{l-1}} \rightarrow \mathbb{R}^{n_l}$ is given by a weight matrix $A^{(l)} \in \mathbb{Z}^{n_l \times n_{l-1}}$ and a bias vector $b^{(l)} \in \mathbb{R}^{n_l}$:

$$\rho^{(l)}(\nu^{(l-1)}) := A^{(l)}\nu^{(l-1)} + b^{(l)}.$$

The (i, j) th coordinate of $A^{(l)}$ will be denoted $a_{ij}^{(l)}$ and the i th coordinate of $b^{(l)}$ by $b_i^{(l)}$. Collectively they form the parameters of the l th layer

This way forward propagation corresponds to evaluation of the neural network.

Remark 2.13. [ZNL18] For a vector input $x \in \mathbb{R}^{n_l}$, $\sigma^{(l)}(x)$ is understood to be in coordinatewise sense; so $\sigma : \mathbb{R}^{n_l} \rightarrow \mathbb{R}^{n_l}$. We assume the final output of a neural network $\nu(x)$ is fed into a score function $s : \mathbb{R}^p \rightarrow \mathbb{R}^m$ that is application specific.

Now we will give a brief introduction to backpropagation of neural networks. The underlying analytics for the small section of backpropagation will not be introduced extensively, rather they'll be required as prerequisite knowledge from the reader.

Backpropagation is best explained by viewing a neural feedforward neural network as a function or a concatenation of functions, which we can extend to a Cost function and perform gradient descent on. In the following we may refer to $A^{(l)}$ as $w^{(l)}$ the weight matrices and $b^{(l)}$ simply as the bias vectors.

Remark 2.14. For a multi-variate function $F(X)$ that is defined and differentiable in a neighbourhood of a point a , then F increases in point a fastest in value in the direction of the gradient $\Delta F(a)$.

This motivates the following definition.

Definition 2.15. (Gradient descent) From remark 2 follows for a defined, differentiable function $F : \mathbb{R} \rightarrow \mathbb{R}$ if

$$a_{n+1} = a_n - \epsilon \Delta F(a_n)$$

for $\epsilon \in \mathbb{R}_+$ small enough and $a_n \in R$. Then $F(a_n) \geq F(a_{n+1})$. With this observation in mind, one starts with a guess $x_0 \in \mathbb{R}$ for a local minimum of F , defines the sequence $(x_n)_{n \in \mathbb{N}}$ as above ($c_{n+1} = c_n - \epsilon \Delta F(x_n)$) and with epsilon small enough x_n will eventually end up close to a local minimum. Setting up and calculating the sequence x_n to step towards local minima is called gradient descent.

We will use gradient descent with a cost function that we want to optimize the neural network on. To optimize a feedforward neural network to give sensible outputs relative to the inputs, we define the cost function.

Definition 2.16. (Cost function) Let $\nu : \mathbb{R}^d \rightarrow \mathbb{R}^p$ be a L -layer feedforward neural network. The cost function for ν to a specific input x and desired output y is C_x or $C_x(\nu(x), y)$ being a differentiable extension of ν dependant on the weights and biases of ν . Notice that the input-output pair (x, y) are fixed, where the weights $w^{(l)}$ and biases $b^{(l)}$ are variable.

The cost function of a finite set \mathcal{X} of input vectors is then the average

$$C = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} C_x$$

over cost functions C_x for individual training examples, x .

Backpropagation is an algorithm to tweak weights of a neural network to minimize the cost function on a training set, containing input vectors and the corresponding desired outputs.

For backpropagation only we limit the σ 's of feedforward neural networks to be differentiable.

Definition 2.17. (Backpropagation) Let ν be an L -layer feedwordward neural network, $\mu \in \mathbb{R}$ a learning rate and C be a cost function over ν as in 2.16. Application of backpropagation to ν is calculating $\Delta_{w^{(i)}} C$ the gradient of the weights and $\Delta_{b^{(i)}} C$ the gradient of the biases of the cost function for $i = 1, \dots, L$ and applying changes to the weights and biases of the neural network ν as follows:

$$\begin{aligned} w^{(i)} &\rightarrow (w^{(i)})' = w^{(i)} - \mu \Delta_{w^{(i)}} C \\ b^{(i)} &\rightarrow (b^{(i)})' = b^{(i)} - \mu \Delta_{b^{(i)}} C \text{ for } i = 1, \dots, L. \end{aligned}$$

Repeated application of backpropagation is a gradient descent algorithm and therefore optimize the cost function and approaches local minima. This gives a way to train a neural network on a datapool efficiently.

Remark 2.18. Essentially, backpropagation evaluates the expression for the derivative of the cost function as a product of derivatives between each layer from left to right – ”backwards” – with the gradient of σ ’s between each layer being a simple modification of the partial products (the ”backwards propagated error”). Given an input–output pair (x, y) , the loss is:

$$C_x(\sigma^{(L)} \circ \rho^{(L)} \circ \sigma^{(L-1)} \circ \rho^{(L-1)} \circ \dots \circ \sigma^{(1)} \circ \rho^{(1)}(x), y).$$

The gradient Δ is the transpose of the derivative of the output in terms of the input and with the chain rule the following holds.

$$\begin{aligned} \frac{dC_x}{dx} &= \frac{dC_x}{d\sigma^{(L)}} \cdot \frac{d\sigma^{(L)}}{d\rho^{(L)}} \cdot \frac{d\rho^{(L)}}{d\sigma^{(L-1)}} \cdot \dots \cdot \frac{d\sigma^{(1)}}{d\rho^{(1)}} \cdot \frac{d\rho^{(1)}}{dx} \\ \Delta_x C_x &= \left(\frac{dC}{dx} \right)^t \end{aligned}$$

The affine linear functions $\rho^{(l)}$ for $l = L, \dots, 1$ can be written unambiguously as it’s weight matrix $w^{(l)}$ and bias vector $b^{(l)}$, $\rho^{(l)} = x^{(l)}x + b^{(l)}$. Each derivative

$$\frac{d\rho^{(l)}}{d\sigma^{(l-1)}}, \text{ for } l = L, \dots, 2$$

therefore consists of the derivative

$$\frac{d\rho^{(l)}}{d\sigma^{(l-1)}} = \frac{dw^{(l)}}{db^{(l)}} \cdot \frac{db^{(l)}}{\sigma^{(l-1)}}.$$

The same holds for $l = 1$ in relation to x . We get

$$\frac{dC_x}{dx} = \frac{dC_x}{d\sigma^{(L)}} \cdot \frac{d\sigma^{(L)}}{dw^{(L)}} \cdot \frac{dw^{(L)}}{db^{(L)}} \cdot \frac{db^{(L)}}{d\sigma^{(L-1)}} \cdot \dots \cdot \frac{d\sigma^{(1)}}{dw^{(1)}} \cdot \frac{dw^{(1)}}{db^{(1)}} \cdot \frac{db^{(1)}}{dx}.$$

This way we set

$$\begin{aligned} \delta_w^{(L)} &= \frac{dC}{d\sigma^{(L)}} \cdot \frac{d\sigma^{(L)}}{dw^{(L)}} \\ \delta_b^{(L)} &= \delta_w^{(L)} \cdot \frac{dw^{(L)}}{db^{(L)}} \\ \delta_w^{(l-1)} &= \delta_b^{(l)} \frac{db^{(l)}}{d\sigma^{(l)}} \cdot \frac{d\sigma^{(l)}}{db^{(l-1)}}, \text{ and} \\ \delta_b^{(l-1)} &= \delta_w^{(l-1)} \cdot \frac{dw^{(l-1)}}{b^{(l-1)}} \text{ for } l = L-1, \dots, 2 \\ \delta^{(0)} &= \delta_b^{(1)} \cdot \frac{db^{(1)}}{x} \end{aligned}$$

Which depending on the choice of sigma gives an easy to calculate algorithm for

$$\begin{aligned} \Delta_{w^{(l)}} C_x &= (\delta_w^{(l)})^t, \text{ and} \\ \Delta_{b^{(l)}} C_x &= (\delta_b^{(l)})^t \text{ for } l = 1, \dots, L. \end{aligned}$$

In contrast to calculating the δ 's by forward propagating.

$$\begin{aligned}\delta^{(0)} &= \frac{dC_x}{d\sigma^{(L)}} \cdot \frac{d\sigma^{(L)}}{dw^{(L)}} \cdot \frac{dw^{(L)}}{db^{(L)}} \cdot \frac{db^{(L)}}{d\sigma^{(L-1)}} \cdot \dots \cdot \frac{d\sigma^{(1)}}{dw^{(1)}} \cdot \frac{dw^{(1)}}{db^{(1)}} \cdot \frac{db^{(1)}}{dx} \\ &\vdots \\ \delta_w^{(L)} &= \frac{dC}{d\sigma^{(L)}} \cdot \frac{d\sigma^{(L)}}{dw^{(L)}}.\end{aligned}$$

This gives a way to train neural networks on a datapool.

Finally for a tropical characterisation of neural networks we will need to restrict the neural networks in the following ways.

Remark 2.19. [ZNL18] We will make the following mild assumptions on the architecture of our feedforward neural networks:

- (a) the weight matrices $A^{(1)}, \dots, A^{(L)}$ are integer-valued;
- (b) the bias vectors $b^{(1)}, \dots, b^{(L)}$ are real-valued;
- (c) the activation functions $\sigma^{(1)}, \dots, \sigma^{(L)}$ take the form

$$\sigma^{(l)}(x) := \max\{c, t^{(l)}\},$$

where $t^{(l)} \in (\mathbb{R} \cup \{-\infty\})^{n_l}$ is called a threshold vector.

Henceforth all neural networks in our subsequent discussions will be assumed to satisfy (a)–(c).

Remark 2.20. [ZNL18] (b) is completely general but there is also no loss of generality in (a), i.e., in restricting the weights A^1, \dots, A^L from real matrices to integer matrices, as:

- real weights can be approximated arbitrarily closely by rational weights;
- one may then ‘clear denominators’ in these rational weights by multiplying them by the least common multiple of their denominators to obtain integer weights;
- keeping in mind that scaling all weights and biases by the same positive constant has no bearing on the workings of a neural network.

The following chapter holds the key theorems of this thesis.

3 Tropical algebra of neural networks

This chapter proves the key theorems of the thesis, an equivalence of neural networks under assumptions (a)-(c) and tropical rational functions, which allows studies on tropical rational functions in return to enforce statements on neural networks.

Remark 3.1. [ZNL18] Consider the output from the first layer in a neural network

$$\nu(x) = \max\{Ax + b, t\};$$

where $A \in \mathbb{Z}^{p \times d}$, $b \in \mathbb{R}^p$, and $t \in (\mathbb{R} \cup \{-\infty\})^p$. We will decompose A as a difference of two nonnegative integervalued matrices, $A = A_+ - A_-$ with $A_+, A_- \in \mathbb{N}^{p \times d}$; e.g., in the standard way with entries

$$a_{ij}^+ := \max\{a_{ij}, 0\}, \quad a_{ij}^- := \max\{-a_{ij}, 0\}$$

respectively. Since

$$\max\{Ax + b, t\} = \max\{A_+x + b, A_-x + t\} - A_-x$$

and

$$\max\{A_+x + b, A_-x + t\} - A_-x = \begin{pmatrix} \max\{a_{1,1}x_1 + \cdots + a_{1d}x_d + b_1\} - \max\{t_1, -\infty\} \\ \vdots \\ \max\{a_{p,1}x_1 + \cdots + a_{pd}x_d + b_p\} - \max\{t_p, -\infty\} \end{pmatrix},$$

we see that every coordinate of one-layer neural network is a difference of two tropical polynomials, which means the first layer of a neural network can be described through a tropical rational map.

For networks with more layers, we apply this decomposition recursively to obtain the following result.

Theorem 3.2. [ZNL18] (Tropical characterization of neural networks). A feedforward neural network under assumptions (a)-(c) is a function $\nu : \mathbb{R}^d \rightarrow \mathbb{R}^p$ whose coordinates are tropical rational functions of the input, i.e.,

$$\nu(x) = F(x) \oslash G(x) = F(x) - G(x)$$

where F and G are tropical polynomial maps. Thus ν is a tropical rational map.

Proof. Let $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{R}^m$ be the parameters of the $(l+1)$ th layer, and let $t \in (\mathbb{R} \cup -\infty)^m$ be the threshold vector in the $(l+1)$ th layer. If the nodes of the l th layer are given by a tropical rational map,

$$\nu^{(l)}(x) = F^{(l)}(x) \oslash G^{(l)}(x) = F^{(l)}(x) - G^{(l)}(x),$$

i.e., each coordinate of $f^{(l)}$ and $G^{(l)}$ is a tropical polynomial in x . We want to show, that then $\nu^{(l+1)}$ is also a rational map. We begin by calculating

$$\begin{aligned}
\rho^{(l+1)} \circ \nu^{(l)}(x) &= A^{(l+1)}\nu^{(l)} + b^{(l)} \\
&= A^{(l+1)}(F^{(l)}(x) - G^{(l)}(x)) + b^{(l+1)} \\
&= (A_+^{(l+1)} - A_-^{(l+1)})(F^{(l)}(x) - G^{(l)}(x)) + b^{(l+1)} \\
&= A_+^{(l+1)}F^{(l)}(x) + A_-^{(l+1)}G^{(l)}(x) + b^{(l+1)} - (A_+^{(l+1)}G^{(l)}(x) + A_-^{(l+1)}F^{(l)}(x)) \\
&:= H^{(l+1)}(x) - G^{(l+1)}(x),
\end{aligned}$$

with

$$\begin{aligned}
H^{(l+1)}(x) &:= A_+F^{(l)}(x) + A_-G^{(l)}(x) + b, \\
G^{(l+1)}(x) &:= A_+G^{(l)}(x) + A_-F^{(l)}(x).
\end{aligned}$$

is a rational map, since when we write $g_i^{(l)}$ and $h_i^{(l)}$ for the i th coordinate of $G^{(l)}$ and $H^{(l)}$ respectively. In tropical arithmetic, the recurrence above takes the form

$$\begin{aligned}
g_i^{(l+1)} &= [\odot_{j=1}^n (g_j^{(l)})^{a_{ij}^+}] \odot [\odot_{j=1}^n (f_j^{(l)})^{a_{ij}^-}] \in \mathbb{T}(x), \\
h_i^{(l+1)} &= [\odot_{j=1}^n (f_j^{(l)})^{a_{ij}^+}] \odot [\odot_{j=1}^n (g_j^{(l)})^{a_{ij}^-}] \odot b_i \in \mathbb{T}(x),
\end{aligned}$$

On this basis it is easy to show that ν

$$\begin{aligned}
\nu^{(l+1)}(x) &= \sigma^{(l+1)} \circ \rho^{(l+1)} \circ \nu^{(l)}(x) \\
&= \sigma^{(l+1)}(H^{(l+1)}(x) - G^{(l+1)}(x)) \\
&= \max\{H^{(l+1)}(x) - G^{(l+1)}(x), t\} \\
&= \max\{H^{(l+1)}(x), t + G^{(l+1)}(x)\} - G^{(l+1)}(x) \\
&:= F^{(l+1)}(x) - G^{(l+1)}(x),
\end{aligned}$$

with

$$F^{(l+1)}(x) := \max\{H^{(l+1)}(x), t + G^{(l+1)}(x)\}$$

is a tropical rational map considering, with f_i denoting the i th coordinate of $F^{(l)}$

$$f_i^{(l+1)} = h_i^{(l+1)} \oplus (g_i^{(l+1)} \odot t_i) \in \mathbb{T}(x)$$

$f_i^{(l+1)}$ is a tropical rational function.

We have shown the induction step, that with Remark 3.1 as induction beginning ends the proof. \square

Remark 3.3. [ZNL18] Note that the tropical rational functions above have real coefficients, not integer coefficients. The integer weights $A^{(l)} \in \mathbb{Z}^{n_l \times n_{l-1}}$ have gone into the powers of tropical monomials in f and g , which is why we require our weights to be integer-valued, although as we have explained, this requirement imposes little loss of generality

By setting $t^{(1)} = \dots = t^{(L-1)} = 0$ and $t^{(L)} = -\infty$, we obtain the following corollary.

Corollary 3.4. [ZNL18] Let $\nu : \mathbb{R}^d \rightarrow \mathbb{R}$ be an ReLU activated feedforward neural network with integer weights and linear output. Then ν is a tropical rational function.

A more remarkable fact is the converse of Corollary 3.4.

Theorem 3.5. [ZNL18] (Equivalence of neural networks and tropical rational functions).

- (i) Let $\nu : \mathbb{R}^d \rightarrow \mathbb{R}$. Then ν is a tropical rational function if and only if ν is a feedforward neural network satisfying assumptions (a)–(c).
- (ii) A tropical rational function $f \odot g$ can be represented as an L -layer neural network, with

$$L \leq \max\{\lceil \log_2 r_f \rceil, \lceil \log_2 r_g \rceil\} + 2,$$

where r_f and r_g are the number of monomials in the tropical polynomials f and g respectively.

Proof. That a feedforward neural network under assumptions (a)–(c) is a tropical polynomial map has been shown in Theorem 3.2. Now with ReLU activations we want to show the opposite implication. As with 3.2, we will show this by induction. Setting $\sigma_t := \max\{x, t\}$ our base case is $\nu(x) = b_i x^{\alpha_i}$ a tropical monomial. Considering

$$b_i x^{\alpha_i} = (\sigma_{-\infty} \circ \rho_i)(x) = \max\{a_i^t x + b_i, -\infty\}$$

ν is a feedforward neural network under assumptions (a)–(c). As induction step we observe two tropical polynomials p and q which are represented as neural networks with l_p and l_q layers respectively,

$$\begin{aligned} p(x) &= (\sigma_{-\infty} \circ \rho_p^{(l_p)} \circ \sigma_0 \circ \dots \circ \sigma_0 \circ \rho_p^{(1)})(x), \\ q(x) &= (\sigma_{-\infty} \circ \rho_q^{(l_q)} \circ \sigma_0 \circ \dots \circ \sigma_0 \circ \rho_q^{(1)})(x) \end{aligned}$$

Here all sigma are set to σ_0 except the last one to $\sigma_{-\infty}$ as prerequisite. We will write $p \oplus q$ as a neural network.

$$\begin{aligned} (p \oplus q)(x) &= \min\{p(x), q(x)\} \\ &= \sigma_{-\infty}(\min\{p(x), q(x)\}) \\ &= \sigma_{-\infty}(\min(p(x) - q(x), 0) + \min(q(x), 0) - \min(-q(x), 0)) \\ &= \sigma_{-\infty}\left(\begin{pmatrix} 1 & 1 & -1 \end{pmatrix} \sigma_0(\rho(y(x))) + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\right) \end{aligned}$$

where $y : \mathbb{R}^d \rightarrow \mathbb{R}^2$ is given by $y(x) = (p(x), q(x))$ and $\rho_i : \mathbb{R}^2 \rightarrow \mathbb{R}$, $i = 1, 2, 3$, are linear functions defined by

$$\rho_1(y) = y_1 - y_2, \rho_2(y) = y_2, \rho_3(y) = -y_2.$$

The function $y(x)$ is a $\max\{l_p, l_q\}$ layer feedforward neural network. This is easily seen by extending the shorter neural network p or q with identity ρ 's, say w.l.o.g. $l_p < l_q$, then $\rho_p^{(i)} = id$ for $i = l_p + 1, \dots, l_q$. Then $y(x) = (\sigma_{-\infty} \circ \rho^{(l_p)} \circ \sigma_0 \circ \dots \circ \sigma_0 \circ \rho^{(1)})(x)$ with $\rho^{(i)}(x) = (\rho_p^{(i)}(x), \rho_q^{(i)}(x))$. By removing the last irrelevant sigma $\sigma_{-\infty}$ off of p and q and since composition of affine linear functions are affine linear, $(p \oplus q)$ is a neural network with $\max\{l_p, l_q\} + 1$ layers. Thus, by induction, any tropical polynomial can be written as a neural network with ReLU activation.

Observe also that if a tropical polynomial is the tropical sum of r monomials, then it can be written as a neural network with no more than $\lceil \log_2 r \rceil + 1$ layers. Now one more time we will write $p \odot q$, where p and q are tropical polynomials as a neural network. Under the same assumptions as above and with

$$(p \odot q)(x) = \sigma_0(p(x)) - \sigma_0(-p(x)) + \sigma_0(-q(x)) - \sigma_0(q(x)) = \sigma_{-\infty}((1 \quad -1 \quad 1 \quad -1) \sigma_0(\rho(y(x))))$$

where $\rho_i : \mathbb{R}^2 \rightarrow \mathbb{R}^1$, $i = 4, 5, 6, 7$, are linear functions defined by

$$\rho_4(y) = y_1, \rho_5(y) = -y_1, \rho_6(y) = -y_2, \rho_7(y) = y_2.$$

The same argumentation as above shows, $p \odot q$ is also a neural network with at most $\max\{l_p, l_q\}$ layers.

Finally, if f and g are tropical polynomials that are respectively tropical sums of r_f and r_g monomials, then the discussions above show that $(f \odot g) = f(x) - g(x)$ is a neural network with at most $\max\{\lceil \log_2 r_f \rceil + 1, \lceil \log_2 r_g \rceil + 1\} + 1 = \max\{\lceil \log_2 r_f \rceil, \lceil \log_2 r_g \rceil\} + 2$ layers. □

Proposition 3.6. [ZNL18] Let $\nu : \mathbb{R}^d \rightarrow \mathbb{R}$. Then ν is a continuous piecewise linear function with integer coefficients if and only if ν is a tropical rational function.

Proof. □

Remark 3.7. [ZNL18] Corollary 5.3, Theorem 5.4, and Proposition 5.5 collectively imply the equivalence of

- (i) tropical rational functions,
- (ii) continuous piecewise linear functions with integer coefficients,
- (iii) neural networks satisfying assumptions (a)-(c).

Proposition 3.8. [ZNL18] Every feedforward neural network with ReLU activation is a tropical rational signomial map.

Proof. □

4 Tropical hypersurfaces

In this chapter we will start viewing tropical polynomials as geometric figures with a surface area that is piecewise linear and boundaries for the number of the linear areas. We hop right in by defining the tropical hypersurface of a tropical polynial.

Definition 4.1. [ZNL18, p. 3] The tropical hypersurface of a tropical polynomial $f(x) = c_1x^{\alpha_1} \oplus \dots \oplus c_rx^{\alpha_r}$ is

$$\Gamma(f) := \{x \in \mathbb{R}^d : c_ix^{\alpha_i} = c_jx^{\alpha_j} = f(x) \text{ for some } \alpha_i \neq \alpha_j\}$$

i.e., the set of points x at which the value of f at x is attained by two or more monomials in f .

To understand the geometry of tropical rational functions, tropical polynomial maps and tropical rational maps better we will show that each are piecewise linear.

Lemma 4.2. Tropical rational functions are piecewise linear functions. Tropical polynomial maps and tropical rational maps are piecewise linear maps.

Proof. Because of 1.13 tropical polynomials are piecewise linear. \odot is a linear transformation, therefore the tropical rational maps are piecewise linear two. Now this means that tropical rational maps and tropical polynomial maps are also piecewise linear and the notion of linear regions applies. \square

Definition 4.3. A linear region of $F \in \text{Rat}(d, m)$ is a maximal connected subset of the domain on which F is linear. The number of linear regions of F is denoted $\mathcal{N}(f)$ [ZNL18, p. 4].

The linearity of a tropical polynomial per definition will be broken on its hypersurface and linear everywhere else. Therefore the tropical hypersurface $\Gamma(f)$ of a tropical polynomial f divides the domain of f into convex cells on each of witch f is linear.

Definition 4.4. [ZNL18, p. 3] Tropical hypersurfaces of polynomials in two variables are called tropical curves.

Definition 4.5. [ZNL18, p. 3] The Newton polygon of a tropical polynomial $f(x) = c_1x^{\alpha_1} \oplus \dots \oplus c_rx^{\alpha_r}$ is the convex hull of $\alpha_1, \dots, \alpha_r \in \mathbb{N}^d$, regarded as points in \mathbb{R}^d ,

$$\Delta(f) := \text{Conv}\alpha_i \in \mathbb{R}^d : c_i \neq -\infty, i = 1, \dots, r$$

.

Remark 4.6. [ZNL18, p. 3] A tropical polynomial f determines a dual subdivision of $\delta(f)$, constructed as follows. First, lift each α_i from \mathbb{R}^d into \mathbb{R}^{d+1}

by appending c_i as the last coordinate. Denote the convex hull of the lifted $\alpha_1, \dots, \alpha_r$ as

$$\mathcal{P}(f) := \text{Conv}(\alpha_i, c_i) \in \mathbb{R}^d \times \mathbb{R} : i = 1, \dots, r. (1)$$

Next let $UF((P)(f))$ denote the collection of upper faces in $\mathcal{P}(f)$ and $\pi : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$ be the projection that drops the last coordinate. The dual subdivision determined by f is then

$$\delta(f) := \pi(p) \subset \mathbb{R}^d : p \in UF(\mathcal{P}(f)).$$

. $\delta(f)$ forms a polyhedral complex with support $\delta(f)$. By [MS15], the tropical hypersurface $\mathcal{T}(f)$ is the $(d-1)$ -skeleton of the polyhedral complex dual to $\delta(f)$. This means that each vertex in $\delta(f)$ corresponds to one "cell" in \mathbb{R}^d where the function f is linear. Thus, the number of vertices in $\mathcal{P}(f)$ provides an upper bound on the number of linear regions of f .

Our analysis of neural networks will require figuring out how the polytope $(P)(f)$ transforms under tropical power, sum, and product.

Proposition 4.7. [ZNL18, p. 4] Let f be a tropical polynomial and let $a \in \mathbb{N}$. Then

$$\mathcal{P}(f^a) = a\mathcal{P}(f)$$

. $a\mathcal{P}(f) = \{ax : x \in \mathcal{P}(f)\} \subset \mathbb{R}^{d+1}$ is a scaled version of $\mathcal{P}(f)$ with the same shape but different volume.

Proof. Let $f(x) = c_1 x^{\alpha_1} \oplus \dots \oplus c_r x^{\alpha_r}$, $f : \mathbb{T}^n \rightarrow \mathbb{T}$, as in 1.8. Then with $\mathcal{P}(f) = \text{Conv}\{(\alpha_i, c_i) \in \mathbb{R}^d \times \mathbb{R}, i = 1, \dots, r \text{ and } c_i \neq -\infty\}$.

$$\begin{aligned} f^a &= (c_1 x^{\alpha_1} \oplus \dots \oplus c_r x^{\alpha_r})^a \\ &= a \cdot (c_1 x^{\alpha_1} \oplus \dots \oplus c_r x^{\alpha_r}) \\ &= a \cdot (c_1 x^{\alpha_1}) \oplus \dots \oplus a \cdot (c_r x^{\alpha_r}) \\ &= (a \cdot c_1) x^{a \cdot \alpha_1} \oplus \dots \oplus (a \cdot c_r) x^{a \cdot \alpha_r} \end{aligned}$$

and therefore

$$\mathcal{P}(f^a) = \text{Conv}\{(a \cdot \alpha_i, a \cdot c_i) \in \mathbb{R}^d \times \mathbb{R}, i = 1, \dots, r \text{ and } c_i \neq -\infty\} = a\mathcal{P}(f)$$

.

□

A generalisation of normal summation, now on sets, is the Minkowski sum.

Definition 4.8. [ZNL18, p. 4] The Minkowski sum of two sets P_1 and P_2 in \mathbb{R}^d is the set

$$P_1 + P_2 := \{x_1 + x_2 \in \mathbb{R}^d : x_1 \in P_1, x_2 \in P_2\}$$

; and for $\lambda_1, \lambda_2 \geq 0$, their weighted Minkowski sum is

$$\lambda_1 P_1 + \lambda_2 P_2 := \{\lambda_1 x_1 + \lambda_2 x_2 \in \mathbb{R}^d : x_1 \in P_1, x_2 \in P_2\}$$

.

One important example for an Minkowski sum is a zonotope. For that we need a notion of the following.

Definition 4.9. A subset $A \subset \mathbb{V}$ of a vector field \mathbb{V} is a line segment, if $A = \{\lambda x + \lambda(1 - y) \mid \lambda \in [0, 1] \subset \mathbb{K}\}$ with \mathbb{K} being the scalar field of \mathbb{V} .

Definition 4.10. [ZNL18, p. 4] The Minkowski sum of line segments is called a zonotope.

Definition 4.11. Let $A \subset \mathbb{R}^n$ be a set. A is dot symmetric around $b \in A$ if for $a \in A$ follows $a + 2(b - a) = 2b - a \in A$

Obviously for A and b as in 4.11 and A is convex, then $b \in A$ the symmetry point must lie in A . We get an intuition of zonotopes by proving the following statement.

Lemma 4.12. A zonotope over \mathbb{R}^n is dot symmetric and convex.

Proof. We proof a zonotope is dot symmetric and convex by induction. As base case let $P \subset \mathbb{R}^n$ be a line segment in between $x, y \in \mathbb{R}^n$ then P is obviously dot symmetric around $b = 0.5(x + y) \in \mathbb{R}^n$ and convex per definition. For the induction step let $A \subset \mathbb{R}^n$ be a convex set and dot symmetric around $b \in \mathbb{R}^n$ and also let $P = \{\lambda x + (1 - \lambda)y \mid \lambda \in [0, 1] \subset \mathbb{R}^n\}$ for $x, y \in \mathbb{R}^n$. Then the Minkowski sum of the two sets is

$$\begin{aligned} A + P &= \{a + p \mid a \in A \text{ and } p \in P\} \\ &= \{a + \lambda x + (1 - \lambda)y \mid a \in A \text{ and } \lambda \in [0, 1]\} \end{aligned}$$

It follows that $a_{A+P} \in A + P$ has the form $a_{A+P} = a + \lambda x + (1 - \lambda)y$ for $a \in A$ and $\lambda \in [0, 1]$. We set $b_{A+P} = (0.5)(x + y) + b$. Then

$$\begin{aligned} a_{A+P} + 2(b_{A+P} - a_{A+P}) &= a + \lambda x + (1 - \lambda)y + 2(0.5(x + y) + b - (a + \lambda x + (1 - \lambda)y)) \\ &= 2b - a + (1 - \lambda)x + \lambda y \in A + P \end{aligned}$$

since $2b - a \in A$ and $(1 - \lambda)x + \lambda y \in P$. Therefore a zonotope is symmetric. Similarly for convexity let $p, q \in A + P$ have the shape

$$\begin{aligned} p &= a_p + \lambda_p x + (1 - \lambda_p)y \\ q &= a_q + \lambda_q x + (1 - \lambda_q)y. \end{aligned}$$

For $\mu \in [0, 1] \subset \mathbb{R}^n$ we need to compute

$$\begin{aligned} \mu p + (1 - \mu)q &= \mu(a_p + \lambda_p x + (1 - \lambda_p)y) + (1 - \mu)(a_q + \lambda_q x + (1 - \lambda_q)y) \\ &= \mu a_p + (1 - \mu)a_q + (\mu \lambda_p + (1 - \mu)\lambda_q)x + (\mu(1 - \lambda_p) + (1 - \mu)(1 - \lambda_q))y \\ &= \mu a_p + (1 - \mu)a_q + \Phi x + (1 - \Phi)y \end{aligned}$$

with $\Phi = \Phi(\mu, \lambda_p, \lambda_q) = \mu \lambda_p + \lambda_q - \mu \lambda_q$. If $0 \leq \Phi(\mu, \lambda_p, \lambda_q) \leq 1$, then a zonotope would be convex, since $(1 - \mu)a_q + \Phi x + (1 - \Phi)y \in P$ would hold and $\mu a_p + (1 - \mu)a_q \in A$ holds anyway. So to conclude we show $0 \leq \Phi(\mu, \lambda_p, \lambda_q) \leq 1$. Because of $\mu \lambda_p \geq 0$ and $\lambda_q \geq \mu \lambda_q$ follows $\Phi \geq 0$. Φ is maximal for $\lambda_p = 1$ and $\Phi(\mu, 1, \lambda_q) = \mu + \lambda_q - \mu \lambda_q \leq 1$. \square

Let \mathcal{V} denote the set of vertices of a polytope P . Clearly, the Minkowski sum of two polytopes is given by the convex hull of the Minkowski sum of their vertex sets, i.e., $P_1 + P_2 = \text{Conv}(\mathcal{V}(P_1) + \mathcal{V}(P_2))$. With this observation, the following is immediate.

Proposition 4.13. [ZNL18, p. 4] Let $f, g \in \text{Pol}(d, 1) = \mathbb{T}[x_1, \dots, x_d]$ be tropical polynomials. Then

$$\begin{aligned}\mathcal{P}(f \odot g) &= \mathcal{P}(f) + \mathcal{P}(g), \\ \mathcal{P}(f \oplus g) &= \text{Conv}(\mathcal{V}(\mathcal{P}(f)) \cup \mathcal{V}(\mathcal{P}(g))).\end{aligned}$$

We reproduce below part of [GS93, Theorem 2.1.20] and derive a corollary for bounding the number of vertices on the upper faces of a zonotope.

Theorem 4.14. [GS93]. Let P_1, \dots, P_k be polytopes in \mathbb{R}^d and let m denote the total number of nonparallel edges of P_1, \dots, P_k . Then the number of vertices of $P_1 + \dots + P_k$ does not exceed

$$\sum_{j=0}^{d-1} \binom{m-1}{j}.$$

The upper bound is attained if all P_i 's are zonotopes and all their generating line segments are in general positions.

And now we come to the main corollary of this chapter, core to chapter 5.

Corollary 4.15. [ZNL18, p. 4] Let $\mathcal{P} \in \mathbb{R}^{d+1}$ be a zonotope generated by m line segments P_1, \dots, P_m . Let $\pi : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d$ be the projection. Suppose \mathcal{P} satisfies:

- (i) the generating line segments are in general positions;
- (ii) the set of projected vertices $\{\pi(v) : v \in \mathcal{V}(\mathcal{P})\} \subseteq \mathbb{R}^d$ are in general position.

Then \mathcal{P} has

$$\sum_{j=0}^d \binom{m}{j}$$

vertices on its upper faces. If either (i) or (ii) is violated, then this becomes an upper bound.

Proof. Let V_1 be the vertices of the upper and V_2 the vertices of the lower face of \mathcal{P} in relation to the projection π . As a consequence of (i), P_1, \dots, P_k are nonparallel and with 4.14 follows the number of vertices of \mathcal{P} is

$$n_1 = 2 \sum_{j=0}^d \binom{m-1}{j}.$$

Because a zonotope is convex 4.12 the upper and lower faces of P must already be all faces and $|V_1 \cup V_2| = n_1$ holds. Also because of 4.12, zonotopes are symmetric and therefore $|V_1| = |V_2|$ the upper and lower faces are of same cardinality. Let $P' := \pi(P)$ be the projection of P in \mathbb{R}^d . (ii) would not hold if $\pi(P_1), \dots, \pi(P_2)$ where not in general position and therefore P' is generated by m line segments and P' has

$$n_2 = 2 \sum_{j=0}^{d-1} \binom{m-1}{j}$$

vertices, since the dimension has gone one lower. For any vertex $v \in P$, if $v \in V_1 \setminus V_2 \cup V_2 \setminus V_1$ then v can not be a vertex of P' since all of these get dropped by the projection. On the other hand $v \in (V_1 \cup V_2) \setminus (V_1 \setminus V_2 \cup V_2 \setminus V_1) = V_1 \cap V_2$ can not get dropped by the projection of P and therefore the number of vertices on P' equals $|V_1 \cap V_2|$, thus $n_2 = |V_1 \cap V_2|$ and we can calculate the number of vertices on the upper face of P :

$$\begin{aligned} |V_2| &= 0.5(|V_1 \cup V_2| - |V_1 \cap V_2|) + |V_1 \cap V_2| \\ &= 0.5(n_1 - n_2) + n_2 \\ &= 0.5(2 \sum_{j=0}^d \binom{m-1}{j} - 2 \sum_{j=0}^{d-1} \binom{m-1}{j}) + 2 \sum_{j=0}^{d-1} \binom{m-1}{j} \\ &= \binom{m-1}{d} + 2 \sum_{j=0}^{d-1} \binom{m-1}{j} \\ &= \sum_{j=0}^d (\binom{m-1}{j} + \binom{m-1}{j-1}) \\ &= \sum_{j=0}^d \binom{m}{j} \end{aligned}$$

□

5 Tropical geometry of neural networks

—missing—

References

- [BBV04] Stephen Boyd, Stephen P Boyd, and Lieven Vandenbergh, *Convex optimization*, Cambridge university press, 2004.
- [BP85] Jean Berstel and Dominique Perrin, *Theory of codes*, Academic Press, 1985.
- [Bry61] Arthur E Bryson, *A gradient method for optimizing multi-stage allocation processes*, Proc. Harvard Univ. Symposium on digital computers and their applications, vol. 72, 1961.
- [CMGS10] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber, *Deep, big, simple neural nets for handwritten digit recognition*, Neural computation **22** (2010), no. 12, 3207–3220.
- [GS93] Peter Gritzmann and Bernd Sturmfels, *Minkowski addition of polytopes: computational complexity and applications to gröbner bases*, SIAM Journal on Discrete Mathematics **6** (1993), no. 2, 246–269.
- [Kel60] Henry J Kelley, *Gradient theory of optimal flight paths*, Ars Journal **30** (1960), no. 10, 947–954.
- [Kle09] Paul Klemperer, *A new auction for substitutes: Central-bank liquidity auctions, 'toxic asset' auctions, and variable product-mix auctions*, Journal of the European Economic Association, Forthcoming (2009).
- [Kri14] Nikolai Krivulin, *Tropical optimization problems*, arXiv preprint arXiv:1408.0313 (2014).
- [LMD⁺11] Quoc V. Le, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen, Marc'Aurelio Ranzato, Jeffrey Dean, and Andrew Y. Ng, *Building high-level features using large scale unsupervised learning*, CoRR **abs/1112.6209** (2011).
- [Mas85] Victor P Maslov, *On a new superposition principle for optimization problem*, Séminaire Équations aux dérivées partielles (Polytechnique) (1985), 1–14.
- [Mik05] Grigory Mikhalkin, *Enumerative tropical algebraic geometry in \mathbb{R}^2* , Journal of the American Mathematical Society **18** (2005), no. 2, 313–377.
- [MP43] Warren S McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics **5** (1943), no. 4, 115–133.
- [MS15] Diane Maclagan and Bernd Sturmfels, *Introduction to tropical geometry*, vol. 161, American Mathematical Soc., 2015.

- [OGAJ20] Fateme Olia, Shaban Ghalandarzadeh, Amirhossein Amiraslani, and Sedighe Jamshidvand, *Analysis of linear systems over idempotent semifields*, Mathematical Sciences **14** (2020), no. 2, 137–146.
- [Pal56] Sanford L Palay, *Synapses in the central nervous system*, The Journal of biophysical and biochemical cytology **2** (1956), no. 4, 193.
- [Qui55] Willard V Quine, *A way to simplify truth functions*, The American mathematical monthly **62** (1955), no. 9, 627–631.
- [RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams, *Learning representations by back-propagating errors*, nature **323** (1986), no. 6088, 533–536.
- [WH60] Bernard Widrow and Marcian E Hoff, *Adaptive switching circuits*, Tech. report, Stanford Univ Ca Stanford Electronics Labs, 1960.
- [ZNL18] Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim, *Tropical geometry of deep neural networks*, arXiv preprint arXiv:1805.07091 (2018).