Tropical Geometry of Deep Neural Networks

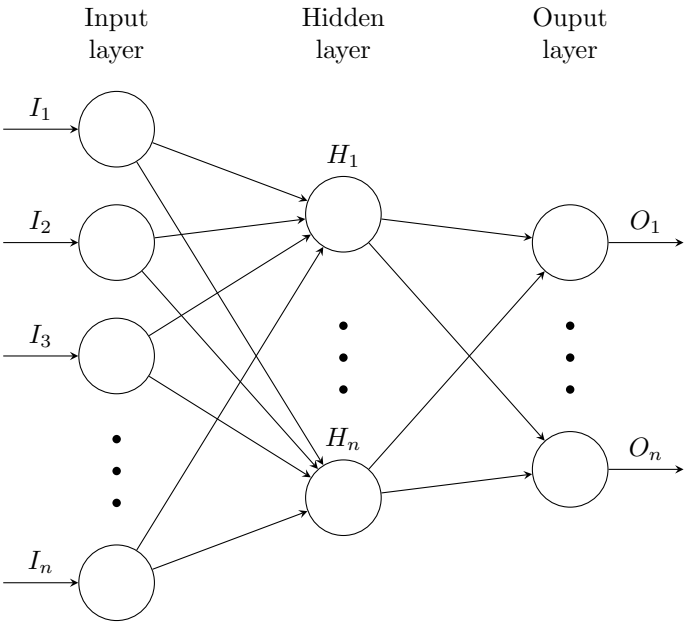
David Leatham

July 28, 2020

Contents

0 Introduction

1 Tropical Algebra



Our basic object of study is the following object $(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$. As a set this is the real numbers $\mathbb{R}$, together with an extra element $-\infty$ whitch represents minus infinity. In this (semiring) the tropical sum of real numbers is ther maximum and the tropical product of real numbers is their usual sum

$$x \oplus y := \max(x,\ y) \quad \& \quad x \odot y := x + y$$

In Tropical Geometry often infinity instead of minus infinity and min in stead of max are used. This does not change any of the underlying theories of tropical algebra, as the the two semirings are tropically Isomorph. Meaning

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

is a tropical Isomorpism.

First we will introduce the the tropical semiring and with it a usual semiring formally.

Definition 1.1. A semiring is a set R equipped with two binary operations $+$ and $\cdot$, called addition and multiplication, such that (Wikipedia.en Semiring)

- $(R, +)$ is a cummutative monoid with identity element 0:
  - $(a + b) + c = a + (b + c)$

- $0 + a = a + 0 = 0$
- $a + b = b + a$

- $(R, \cdot)$ is a monoid with identity element 1:
  - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - $1 \cdot a = a \cdot 1 = a$

- 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. $\mathbb{T} := (\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$ is a semiring called the tropical semiring. (Maclagan & Sturmfels, 2015, p. 10)

Proof.

(1): The neutral element for the tropical sum is $-\infty$ since for $x \in \mathbb{R} \cup \{-\infty\}$ the following stands $x \oplus \infty = \max(x,\ -\infty) = x$ and with $x \odot 0 = x + 0 = x$ for $x \in \mathbb{R}$, 0 is the neutral elemet of tropical multiplication.

(2): Both addition and multiplication are commutative. To prove this we take $x, y \in \mathbb{R} \cup \{-\infty\}$ and do a case distinct. Because $\mathbb{R}$ is a field w.l.o.g. we set $x = -\infty, y \in \mathbb{R}$

$$-\infty \oplus y = \max(-\infty,\ y) = y = \max(y,\ -\infty) = y \oplus \infty$$
$$\infty \odot y = \infty + y = \infty = y + \infty = y \odot \infty$$

(3): Tropical multiplication distributes over addition. Take $x, y, z \in \mathbb{R}$ then

$$x \odot (y \oplus z) = x + \max(y,\ z) = \max(x + y,\ x + z) = (x \odot y) \oplus (x \odot z)$$
$$(y \oplus z) \odot x = \max(y,\ z) + x = \max(y + x,\ z + x) = (y \odot x) \oplus (z \odot x)$$

(4): Multiplication by $-\infty$ annihilates $-\infty \odot x = -\infty\ \forall x \in \mathbb{R} \cup \{-\infty\}$.

□

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. (Maclagan & Sturmfels, 2015, p. 11)

To get more familiar with the tropical arithmetics we will have a look at an arithemtical example, before we jump into tropical polynomials.

Remark 1.3. The tropical Pascal's triangle, whose rows are the coefficients appearing in a binomial expansion, is very simple to remember. All its coefficients are Zero. For example, the fourth row in the triangle is represented be the following equation

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

You may say the Pascal's coefficients are four zeroes. The same applies to all cases

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

And Pascal's triangle has the following form.

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

The goal of this first section and the following is to get to understand tropical rational functions and ¡ maps, as the underlying objects are key to later relations between neural networks and Tropical Geometry. Linking these two field is core to this thesis. First, so that we can introduce multidimensional tropical polynomials properly a notion of monomials is needed.

Definition 1.4. (Zhang, Naitzat, & Lim, 2018, p. 2) A tropical monomial in d variables $x_1, \ldots, x_d$ is an expression or function $\mathbb{T}^d \to \mathbb{T}$ of the from

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

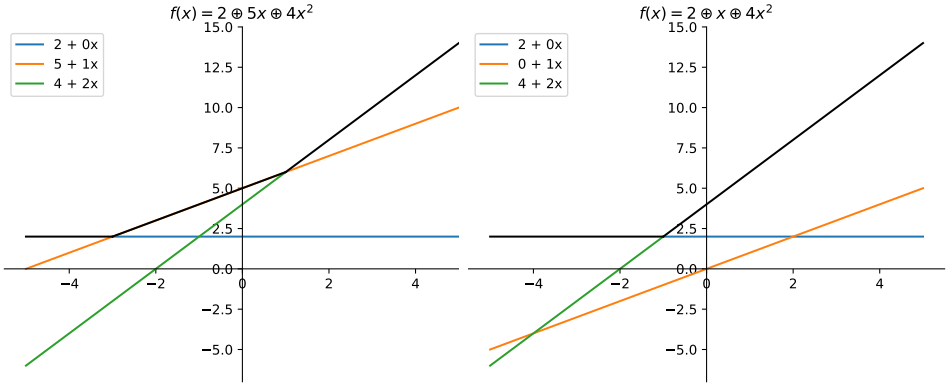
where $c \in \mathbb{R} \cup \{-\infty\}$ and $a_1, \ldots, a_d \in \mathbb{N}$. As a convenient shorthand, we will also write a tropical monomial in multiindex notation as cx_α where $\alpha = (a_1, \ldots, a_d) \in \mathbb{N}_d$ and $x = (x_1, \ldots, x_d)$. Note that $x^\alpha = 0 \odot x^\alpha$.

Definition 1.5. (Zhang et al., 2018, p. 2) Following notations above, a tropical polynomial $f(x) = f(x_1, \ldots, x_d), f : \mathbb{T}^d \to \mathbb{T}$ is a finite tropical sum of tropical monomials

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

where $\alpha_i = (\alpha_{i1}, \ldots, \alpha_{id}) \in \mathbb{N}^d$ and $c_i \in \mathbb{R} \cup \{-\infty\}$, $i = 1, \ldots, r$. 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 $i \neq j$.

Remark 1.6. We are going to examine tropical polynomials in one variable. Monomials in one variable are of the form $c \odot x^a, \mathbb{T} \to \mathbb{T}$ where $c \in \mathbb{T}$ and $a \in \mathbb{N}$. Meaning a tropical polynomial in one variable is of the form $f(x) = c_1 x^{\alpha_1} \oplus \cdots \oplus c_r x^{\alpha_r}$ *with* $c_i \in \mathbb{T}, a_i \in \mathbb{N}$ *for* $i = 1, \ldots, r$.
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 $b - c > a$, otherwise it only breaks at one point $(a - c) : 2$. Visualising two quadratic tropical functions gives a good intuition of the piecewise-linearity.
The black colored parts of the followig figures one and two 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 $5 - 2$, there $5 + x$ dominates to the point $4 + 2x$.

(b) Take $f(x) = 2 \oplus x \oplus 4x^x$, we have changed the scalar in the second part, then 2 dominates, from 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:

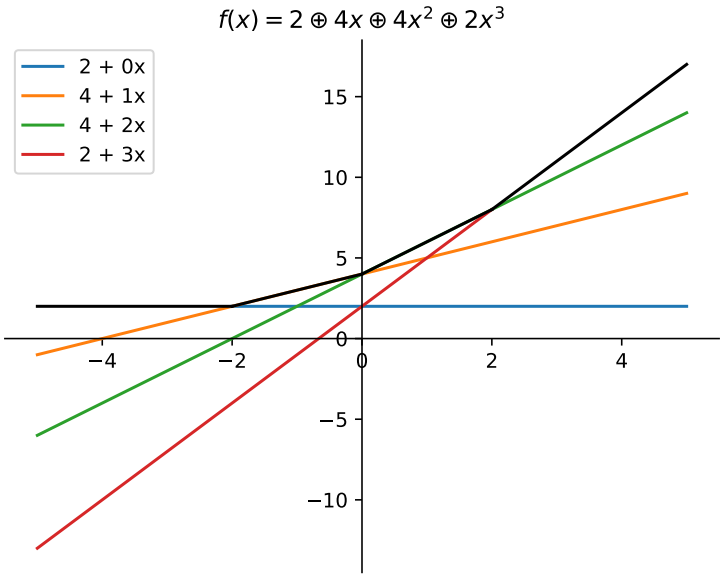


Figure 2: Take $f(x) = 2 \oplus x \oplus 4x^x \oplus 4x^2 \oplus 3x^3$. For a degree three tropical polynomial in one variable this polynomial has reached the maximum number of non linear edges.

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

Lemma 1.7. Let f be a tropical polynomial

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

as in 1.5. 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. (1) The minimum of continuous functions is still continuous.
(2) Every monomial $c_i x^{\alpha_i} = c_i + xy\alpha_{i1} + \cdots + x_r\alpha_{ir}$ 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 $i, j = 1, \ldots, r$.
A piece is a single piece of $c_i x^{\alpha_i}$ where 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.
(3) On one piece f is convex. Moving from one piece to another the slope can only increase, with means f is still convex. □

Proposition 1.8. Every function $\mathbb{R}^n \to \mathbb{R}$ which satisfies the three properties (1), (2) and (3) has a representation as the minimum of a finite set of linear functions. Thus, the tropical polynomial 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 1.7. □

Now we are ready to introduce tropical rational functions. These are important to understand the core section, section 5, in particular the actual connection build between neural networks and tropical geometry in this thesis, as are the semifields $\mathbb{T}[X_1, \dots, X_d]$ and $\mathbb{T}(X_1, \dots, X_d)$.

Definition 1.9. (Zhang et al., 2018, p. 3) Following notations above, 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 1.10. $\mathbb{T}[X_1, \dots, X_d] := \{f : \mathbb{T}^d \to \mathbb{T}; f \textit{ is tropical polynomial}\}$ and $\mathbb{T}(X_1, \dots, X_d) := \{f : \mathbb{T}^d \to \mathbb{T}; f \textit{ is tropical rational function}\}$ are both semifields. (Zhang et al., 2018, p. 3)

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

$$\begin{aligned}
f(x) &= f_1(x) \oslash f_2(x) = (\oplus_{i=0}^{r} c_{1i} x^{\alpha_{1i}}) \oslash (\oplus_{i=0}^{r} c_{2i} x^{\alpha_{2i}}) \\
g(x) &= g_1(x) \oslash g_2(x) = (\oplus_{i=0}^{r} d_{1i} x^{\beta_{1i}}) \oslash (\oplus_{i=0}^{r} d_{2i} x^{\beta_{2i}}) \\
h(x) &= h_1(x) \oslash h_2(x) = (\oplus_{i=0}^{r} e_{1i} x^{\gamma_{1i}}) \oslash (\oplus_{i=0}^{r} e_{2i} x^{\gamma_{2i}})
\end{aligned}$$

We begin the proof by showing, that for two tropical polynomials $a(x) = \oplus_{i=0}^{r} z_i x^{\zeta_i}), b(x) = \oplus_{i=0}^{r} o_i x^{\omega_i}) \in \mathbb{T}[X_1, \dots, X_r]$ the normal sum is a tropical polynomial $(a + b)(x) \in \mathbb{T}[X_1, \dots, X_r]$.

$$\begin{aligned}
(a + b)(x) &= a(x) + b(x) \\
&= (\oplus_{i=0}^{r} z_i x^{\zeta_i}) + (\oplus_{i=0}^{r} o_i x^{\omega_i}) \\
&= \oplus_{i,j=0,\dots,r} (z_i + \zeta_i * x + o_j + \omega_i * x) \\
&= \oplus_{i,j=0,\dots,r} ((z_i + o_j) + x^{\zeta_i + \omega_i}) \in \mathbb{T}[X_1, \dots X_r]
\end{aligned}$$

Other than with the proof of the Tropical semiring we will show that topical tropical addition and tropical multiplication of tropical polynomials as tropical rational functions stay tropical polynomials respectively tropical rational functions. All other axioms stay pointwise the same.

(1): The Tropical sum of a Tropical rational functions is a tropical tropical

rations function

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

Since Addition as tropical addition of tropical polynomials is a tropical polynomial.

(2): The Tropical product of a Tropical rational functions is a tropical tropical rational function

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

(3): The neutral element for the tropical sum is $-\infty = -\infty \oslash x = x \oslash -\infty\ x \in \mathbb{T}$ since for $f(x) \in \mathbb{T}(X_1, \ldots, X_r)$ as above, the following stands $f(x) \oplus \infty = \max(f(x),\ -\infty) = f(x)$ and with $f(x) \odot 0 = f(x) + 0 = f(x) \forall f(x) \in \mathbb{T}\ 0$ is the neutral element of tropical multiplication.

□

Comment 1.11. We regard a tropical polynomial $f = f \oslash 0$ as a special case of a tropical rational function and thus $\mathbb{T}[X_1, \ldots, X_r] \subseteq \mathbb{T}(X_1, \ldots, X_r)$ (Zhang et al., 2018, p. 3).

Comment 1.12.

- A d-variate tropical polynomial $f(x)$ defines a function $f : \mathbb{T}^d \to \mathbb{T}$ that is a convex function in the usual sense as taking max and $\sum$ of convex functions preserve convexity (Boyd, Boyd, & Vandenberghe, 2004).
- As such, a tropical rational function $f \oslash g : \mathbb{T}^d \to \mathbb{T}$ is a DC function or differenceconvex function (Hartman et al., 1959).

Remark 1.13. The only thing missing to this section are some examples for tropical rational functions and tropical rational maps.
!!! Expand by adding some examples !!!

Definition 1.14. $R : \mathbb{R}^d \to \mathbb{R}^p, x = (x_1, \ldots, x_d) \mapsto (f_1(x), \ldots, f_p(x))$, is called a tropical polynomial map if each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a tropical polynomial, $i = 1, \ldots, p$, and a tropical rational map if $f_1, \ldots, f_p$ 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 $Pol(d, 1) = \mathbb{T}[X_1, \ldots, x_d]$ and $Rat(d, 1) = \mathbb{T}(x_1, \ldots, x_d)$ (Zhang et al., 2018, p. 3).

2 Valuations and tropicalisation

Definition 2.1. (Maclagan & Sturmfels, 2015, p. 57) Let $\mathbb{K}$ be a field. We denote by K^* the nonzero elements of K. A valuation on $\mathbb{K}$ is a function $val : \mathbb{K} \to \mathbb{R} \cup \{\infty\}$ satisfying the following three axioms:

1. $val(a) = \infty$ if and only if $a = 0$,
2. $val(ab) = val(a) + val(b)$ and
3. $val(a + b) \geq \min\{val(a), val(b)\}$ for all $a, b \in K^*$

The image of the valuation map is denoted Γ_{val}. This is an additive subgroup of the real numbers $\mathbb{R}$ which is called the value group.

Lemma 2.2. Let $a, b \in \mathbb{K}$. If $val(a) \neq val(b)$ then $val(a+b) = min(val(a), val(b))$. (Maclagan & Sturmfels, 2015, p. 57)

Proof. Let $a, b \in \mathbb{K}$. Without loss of generality we may assume that $val(b) > val(a)$. When evaluating $1 \in \mathbb{K}$ we observe

$$val(1) = val(1^2) = val(1) + val(1)$$

it follows, that $val(1) = 0$, and so $-(1)^2$ implies $val(-1) = 0$. But with the valuation of -1 being zero

$$val(-b) = val(-1) + val(b) = val(b) \text{ for all } b \in \mathbb{K}$$

the sign is irrelevant to valuation. The third axiom implies

$$val(a) \geq \min(a + b), val(-b)) = min(val(a + b), val(b)),$$

and therefore $val(a) \geq val(a + b)$. But we also have

$$val(a + b) \geq \min(val(a), val(b)) = val(a),$$

and hence $val(a + b) = val(a)$ as desired. □

3 Tropical hypersurfaces

Definition 3.1. The tropical hypersurface of a tropical polynomial $f(x) = c_1 x^{\alpha_1} \oplus \cdots \oplus c_r x^{\alpha_r}$ is

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

i.e., the set of points x at which the value of f at x is attained by two or more monomials in f (Zhang et al., 2018, p. 3).

Comment 3.2. 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 $\alpha_1, \ldots, \alpha_r$ as

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

Next let $UF((P)(f))$ denote the collection of upper faces in $\mathcal{P}(f)$ and $\pi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ 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 (Maclagan & Sturmfels, 2015), 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 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.

Definition 3.3. The Newton polygon of a tropical polynomial $f(x) = c_1 x^{\alpha_1} \oplus \cdots \oplus c_r x^{\alpha_r}$ is the convex hull of $\alpha_1, \ldots, \alpha_r \in \mathbb{N}^d$, regarded as points in $\mathbb{R}^d$,

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

(Zhang et al., 2018, p. 3).

Definition 3.4. A linear region of $F \in Rat(d, m)$ is a maximal connected subset of the domain on which F is linear. The number of linear regions of F is denoted $\mathcal{N}(f)$ (Zhang et al., 2018, p. 4).

3.1 Transformations of tropical polynomial

Proposition 3.5. 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 (Zhang et al., 2018, p. 4).

Proof. □

Definition 3.6. 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

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

(Zhang et al., 2018, p. 4).

Proposition 3.7. (Zhang et al., 2018, p. 4) Let $f, g \in Pol(d, 1) = \mathbb{T}[x_1, \ldots, x_d]$ be tropical polynomials. Then

$$\mathcal{P}(f \odot g) = \mathcal{P}(f) + \mathcal{P}(g), \mathcal{P}(f \oplus g) \qquad = Conv(\mathcal{V}(\mathcal{P}(f)) \cup \mathcal{V}(\mathcal{P}(g))).$$

Theorem 3.8. (Gritzmann-Sturmfels). Let $P_1, \ldots, P_k$ be polytopes in $\mathbb{R}^d$ and let m denote the total number of nonparallel edges of $P_1, \ldots, P_k$. Then the number of vertices of $P_1 + \cdots + P_k$ 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. (Gritzmann & Sturmfels, 1993)

Corollary 3.9. Let $\mathcal{P} \in \mathbb{R}^{d+1}$ be a zonotope generated by m line segments $P_1, \ldots, P_m$. Let $\pi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ be the projection. Suppose 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 generalposition.

Then 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. (Zhang et al., 2018, p. 4)

4 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 (McCulloch & Pitts, 1943; Widrow & Hoff, 1960; Rumelhart, Hinton, & Williams, 1986). 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 (McCulloch & Pitts, 1943). Through synapses the axons are connected to further soma through with the impulse is passed on to further neurons. Impulses passing through the nervous system partly consist of electrical impulses and chemical reactions (Palay, 1956). A collection of partially connected neurons, capable of carrying impulses, is called a biological neural network.

Depiction of a neuron

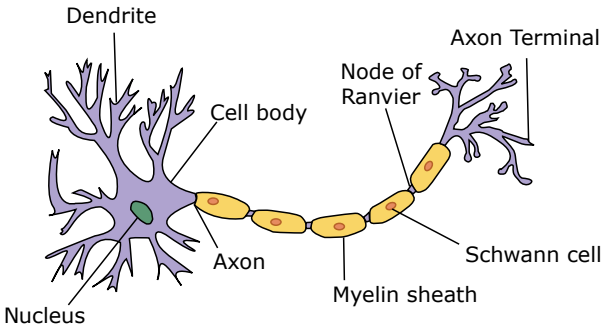


Figure 3: A representation of a neuron. The axon terminal attach to dendrite. This way impulses pass from neuron to neuron.

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 is 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 4.1. 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 4.2. 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 4.3. A weighted graph $G = (V, E)$ in our case is attributed by two functions $\psi : V \to \mathbb{K}$ and $\omega : E \to \mathbb{K}$ 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 4.4. 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 them selves like molecules.

Remark 4.5. 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 cites and we want to find the minimal distance from any city to city A.

Min distance weighted graph

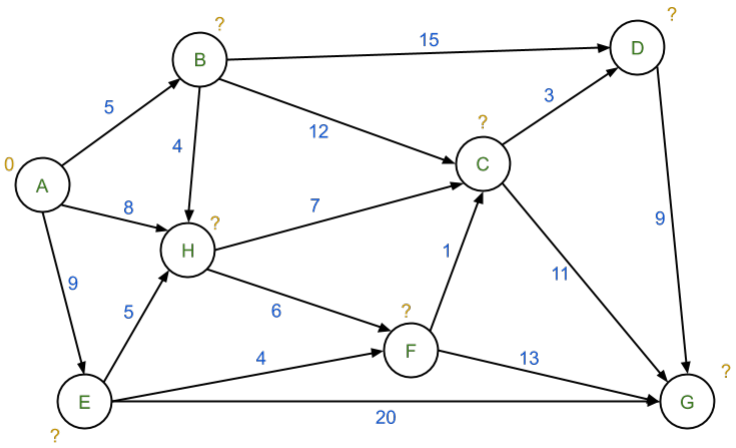


Figure 4: Min distance weighted graph.

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 proof 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 traveling 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. □

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 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 it's length being 14.

One more nice thing to do would be to elaborate and introduce a neural network, that does not have a specific input or out. For example no output. Right now I have no inspiration for this specific topic, so I will go on.

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 4.6. An L-layer feedforward neural network in graph form (G, σ) is a weighted graph $G = (V, E)$ with an activation function σ. The graph consists of L sets $V^{(j)}$, $j = 1, \ldots, L$ of vertices ($V = \cup_{j=1}^{L} V^{(j)}$) with $L-1$ corresponding sets of edges $E^{(i)} \subset \{(x, y)|x \in V^{(i)}; y \in V^{(i+1)}\}$, $i = 1, \ldots, L-1$ 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.

Remark 4.7. 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 4.8. A fully connected feedforward neural network (G, σ) with $G = (V, E)$ is one where E is the largest out of every possible set of E. This will give the unique solution, where $V^{(j)}$ and $V^{(j+1)}$ for each $j = 1, \ldots, L-1$ 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 feed forward neural networks we have cast an sensible form of neural network with 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 4.9. 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 $b^{(j)} = (b_{j1}, \ldots, b_{j|V^{(j)}|})^t$, we call this vector the bias of layer j, with corresponds to the weights of vertices in $V^{(j)}$ for $j = 2, \ldots, L$ and a matrix $W^{(j)}$ with the entries $W^{(j)}_{nm} = \omega((b_{j-1n}, b_{jm}))$ which correspond to the weights of edges connecting layer $j-1$ and j where j is still in range $j = 2, \ldots, L$.

We will label feedforward neural networks this way untill the definition of feedforward neural networks as functions. And are now in a position to define forward propagation.

Definition 4.10. (Forward Propagation) Let (G, σ) be as in 4.9. 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 $j = 2, \ldots, L$ and $a^{(1)} = x$.

Often binary inputs, as for example black and white pictures, are fed in to neural networks and binary outputs, like dog or cat are expected. For the sake of proving, that the feedforward 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.

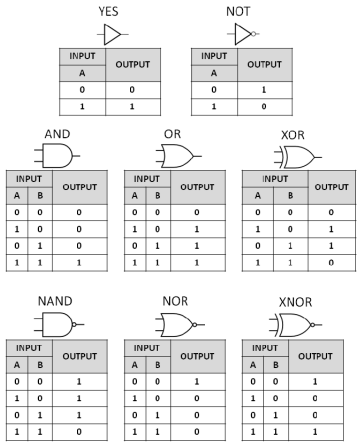
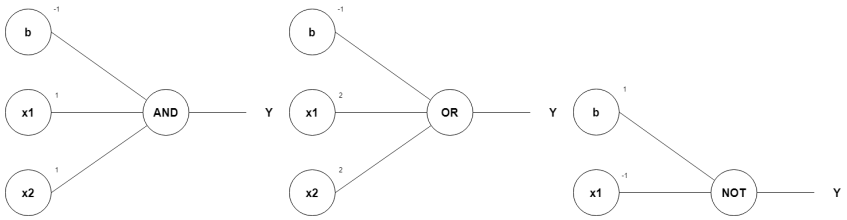


Figure 5: Boolean Gates Truth Tables

Remark 4.11. 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 (Quine, 1955) 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 $\omega = \mathbb{1}$, the weights of edges to one and

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

Usually the output of a neural network is non binary, but is trained so that the higher the output value is, the higher the probabillity of the specific outcome and the lower the output the lower the probabillity, with zero beeing 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.



(a) AND gate (b) OR gate (c) NOT gate

Figure 6: Neural Logical Gates

The first, motivated by figure 6 (a), defies 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

$$\psi(b_{11}) = \psi(b_{12}) = \psi(b_31) = 0,\ \psi(b_{21}) = -1 \ and\ W^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},\ W^{(2)} = \begin{pmatrix} 1 \end{pmatrix}.$$

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

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

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

The same graph with weights

$$\psi(b_{11}) = \psi(b_{12}) = \psi(b_31) = 0,\ \psi(b_{21}) = -1 \ and\ W^{(1)} = \begin{pmatrix} 2 \\ 2 \end{pmatrix},\ W^{(2)} = \begin{pmatrix} 1 \end{pmatrix}$$

makes an OR gate:

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

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

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

$$\psi(b_{11}) = \psi(b_31) = 0,\ \psi(b_{21}) = 1 \textit{ and } W^{(1)} = (-1),\ W^{(2)} = (1)$$

that define a NOT gate:

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

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 abstaction level of feedforward neural networks is to define them as function, witch enables to us to compare feedforward neural networds to tropical polynomials in Chapter 4.

Definition 4.12. Viewed abstractly, an L-layer feedforward neural network is a map $\nu : \mathbb{R}^d \to \mathbb{R}^p$ given by a composition of functions

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

The preactivation functions $\rho^{(1)}, \ldots, \rho^{(L)}$ are affine transformations to be determined and the activation functions $\sigma^{(1)}, \ldots, \sigma^{(L)}$ are chosen and fixed in advanced.
We denote the width, i.e., the number of nodes, of the lth layer by $n_l, l = 1, \ldots, L-1$. We set $n_0 := d$ and $n_L := p$, respectively the dimensions of the input and output of the network. The output from the lth layer will be denoted by

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

i.e., it is a map $\nu^{(l)} : \mathbb{R}^d \to \mathbb{R}^{n_l}$. For convenience, we assume $\nu^{(0)}(x) := x$.
The affine function $\nu^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ 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^l_{ij} and the ith coordinate of $b^{(l)}$ by $b^{(l)}_i$. Collectively they form the parameters of the lth layer

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

Comment 4.13. For a vector input $x \in \mathbb{R}^{n_l}, \sigma^{(l)}(x)$ is understood to be in coordinatewise sense; so $\sigma : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$. We assume the final output of a neural network $\nu(x)$ is fed into a score function $s : \mathbb{R}^p \to \mathbb{R}^m$ 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, with 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 4.14. For a multi-variate function $F(X)$ that is defined and differentiable in a neighborhood 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 4.15. (Gradient descent) From remark 4 follows for a defined, differentiable function $F : \mathbb{R} \to \mathbb{R}$ 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 4.16. (Cost function) Let $\nu : \mathbb{R}^d \to \mathbb{R}^p$ be a L-layer feedworward 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 an 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 differentiable.

Definition 4.17. (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 4.16. 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 $i = 1, \ldots, L$ and applying changes to the to the weights and biases of the neural network ν as follows:

$$w^{(i)} \to (w^{(i)})' = w^{(i)} - \mu\Delta_{w^{(i)}}C$$
$$b^{(i)} \to (b^{(i)})' = b^{(i)} - \mu\Delta_{b^{(i)}}C \text{ for } i = 1, \ldots, L.$$

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.

Comment 4.18. 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:

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

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.

$$\frac{\mathrm{d}C_x}{\mathrm{d}x} = \frac{\mathrm{d}C_x}{\mathrm{d}\sigma^{(L)}} \cdot \frac{\mathrm{d}\sigma^{(L)}}{\mathrm{d}\rho^{(L)}} \cdot \frac{d\rho^{(L)}}{d\sigma^{(L-1)}} \cdot \ \ldots \ \cdot \frac{d\sigma^{(1)}}{d\rho^{(1)}} \cdot \frac{d\rho^{(1)}}{dx}$$
$$\Delta_x C_x = \left(\frac{dC}{dx}\right)^t$$

The affine linear functions $\rho^{(l)}$ for $l = L, \ldots, 1$ 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

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

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

$$\frac{\mathrm{d}C_x}{\mathrm{d}x} = \frac{\mathrm{d}C_x}{\mathrm{d}\sigma^{(L)}} \cdot \frac{\mathrm{d}\sigma^{(L)}}{\mathrm{d}w^{(L)}} \cdot \frac{\mathrm{d}w^{(L)}}{\mathrm{d}b^{(L)}} \cdot \frac{db^{(L)}}{d\sigma^{(L-1)}} \cdot \ \ldots \ \cdot \frac{d\sigma^{(1)}}{dw^{(1)}} \cdot \frac{dw^{(1)}}{db^{(1)}} \cdot \frac{db^{(1)}}{dx}.$$

This way we set

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

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

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

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

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

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

Comment 4.19. We will make the following mild assumptions on the architecture of our feedforward neural networks:

(a) the weight matrices $Aa^{(1)}, \ldots, A^{(L)}$ are integer-valued;

(b) the bias vectors $b^{(1)}, \ldots, b^{(L)}$ are real-valued;

(c) the activation functions $\sigma^{(1)}, \ldots, \sigma^{(L)}$ 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).

5 Tropical algebra of neural networks

Comment 5.1. 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

$$a^+_{ij} := \max\{a_{ij}, 0\},\ a^-_{ij} := \max\{-a_i j, 0\}$$

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, witch 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 5.2. (Tropical characterization of neural networks).
A feedforward neural network under assumptions (a)–(c) is a function $\nu : \mathbb{R}^d \to \mathbb{R}^p$ 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 lth 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

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

with

$$H^{(l+1)}(x) := A_+ F^{(l)}(x) + A_- G(l)(x) + b,$$
$$G^{(l+1)}(x) := A_x G^{(l)}(x) + A_- F^{(l)}(x).$$

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

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

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 ith 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 Comment 5.1 as induction beginning ends the proof. □

Comment 5.3. 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 $t^{(1)} = \cdots = t^{(L-1)} = 0$ and $t^{(l)} = -\infty$, we obtain the following corollary.

Corollary 5.4. Let $\nu : \mathbb{R}^d \to \mathbb{R}$ 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 5.4.

Theorem 5.5. (Equivalence of neural networks and tropical rational functions).

(i) Let $\nu : \mathbb{R}^d \to \mathbb{R}$. 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 $f \oslash g$ 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 5.2. Now with relu activations we want to show the opposite implication. As with 5.2, 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 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 assumtions (a)-(c). As induction step we observe two tropical polynomials p and q with are represented as neural networks with l_p and l_q layers respectively,

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

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.

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

where $y : \mathbb{R}^d \to \mathbb{R}^2$ is given by $y(x) = (p(x), q(x))$ and $\rho_i : \mathbb{R}^2 \to \mathbb{R},\ i = 1, 2, 3,$ 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 $i = l_p + 1, \ldots, l_q$. Then $y(x) = (\sigma_{-\infty} \circ \rho^{(l_p)} \circ \sigma_0 \circ \cdots \circ \sigma_0 \circ \rho^{(1)})(x)$ 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 $p \oslash q$, where p and q are tropical polynomials as a neural network. Under the same assumptions as above and with

$$(p \oslash q)(x) = \sigma_0(p(x)) - \sigma_0(-p(x)) + \sigma_0(-q(x)) - \sigma_0(q(x)) \quad = \sigma_{-\infty}(\begin{pmatrix} 1 & -1 & 1 & -1 \end{pmatrix} \sigma_0(\rho(y(x)))$$

where $\rho_i : \mathbb{R}^2 \to \mathbb{R}^1,\ i = 4, 5, 6, 7$, 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, $p \oslash q$ 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 $(f \oslash g) = f(x) - g(x)$ 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 5.6. Let $\nu : \mathbb{R}^d \to \mathbb{R}$. Then ν is a continuous piecewise linear function with integer coefficients if and only if ν is a tropical rational function.

Comment 5.7. 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 5.8. Every feedforward neural network with ReLU activation is a tropical rational signomial map.

References

Boyd, S., Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Gritzmann, P., & Sturmfels, B. (1993). Minkowski addition of polytopes: computational complexity and applications to gröbner bases. *SIAM Journal on Discrete Mathematics*, *6*(2), 246–269.

Hartman, P., et al. (1959). On functions representable as a difference of convex functions. *Pacific Journal of Mathematics*, *9*(3), 707–713.

Maclagan, D., & Sturmfels, B. (2015). *Introduction to tropical geometry* (Vol. 161). American Mathematical Soc.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Palay, S. L. (1956). Synapses in the central nervous system. *The Journal of biophysical and biochemical cytology*, *2*(4), 193.

Quine, W. V. (1955). A way to simplify truth functions. *The American mathematical monthly*, *62*(9), 627–631.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Widrow, B., & Hoff, M. E. (1960). *Adaptive switching circuits* (Tech. Rep.). Stanford Univ Ca Stanford Electronics Labs.

Zhang, L., Naitzat, G., & Lim, L.-H. (2018). Tropical geometry of deep neural networks. *arXiv preprint arXiv:1805.07091*.