From Complexity to Clarity: Analytical Expressions of Deep Neural Network Weights via Clifford Algebra and Convexity

Mert Pilanci *pilanci@stanford.edu*
Department of Electrical Engineering
Stanford University
www.stanford.edu/∼pilanci

Reviewed on OpenReview: *https://openreview.net/forum?id=6XPwSwEWsV*

Abstract

In this paper, we introduce a novel analysis of neural networks based on geometric (Clifford) algebra and convex optimization. We show that optimal weights of deep ReLU neural networks are given by the wedge product of training samples when trained with standard regularized loss. Furthermore, the training problem reduces to convex optimization over wedge product features, which encode the geometric structure of the training dataset. This structure is given in terms of signed volumes of triangles and parallelotopes generated by data vectors. The convex problem finds a small subset of samples via ℓ_1 regularization to discover only relevant wedge product features. Our analysis provides a novel perspective on the inner workings of deep neural networks and sheds light on the role of the hidden layers.

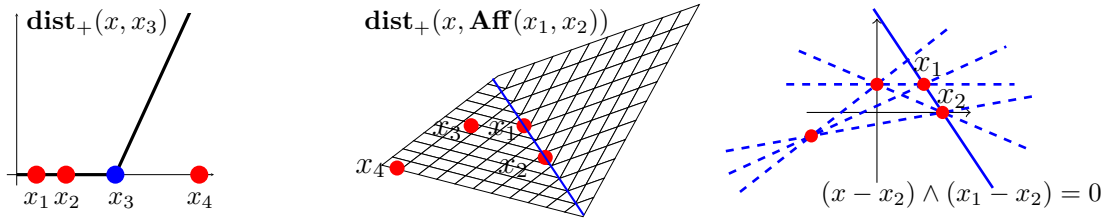
1 Introduction

While there has been a lot of progress in developing deep neural networks (DNNs) to solve practical machine learning problems (Krizhevsky et al., 2012; LeCun et al., 2015; OpenAI, 2023), the inner workings of neural networks is not well understood. A foundational theory for understanding how neural networks work is still lacking despite extensive research over several decades. In this paper, we provide a novel analysis of neural networks based on geometric algebra and convex optimization. We show that weights of deep ReLU neural networks learn the wedge product of a subset of training samples when trained by minimizing standard regularized loss functions. Furthermore, the training problem for two-layer and three-layer networks reduces to convex optimization over wedge product features, which encode the geometric structure of the training dataset. This structure is given in terms of signed volumes of triangles and parallelotopes generated by data vectors. By the addition of an additional ReLU layer, the wedge products are iterated to yield a richer discrete dictionary of wedge features. Our analysis provides a novel perspective on the inner workings of deep neural networks and sheds light on the role of the hidden layers.

1.1 Prior work

The quest to understand the internal workings of neural networks (NNs) has led to numerous theoretical and empirical studies over the years. A striking discovery is the phenomenon of "neural collapse," observed when the representations of individual classes in the penultimate layer of a deep neural network tend to a point of near-indistinguishability (Papyan et al., 2020). Despite this insightful finding, the underlying mechanism that enables this collapse is yet to be fully understood. Linearizations and infinite-width approximations have been proposed to explain the inner workings of neural networks (Jacot et al., 2018; Chizat et al., 2019; Radhakrishnan et al., 2023). However, these approaches often simplify the rich non-linear interactions inherent in deep networks, potentially missing out on the full spectrum of dynamics and behaviors exhibited during training and inference.

Infinite dimensional convex neural networks were introduced in Bengio et al. (2005), offering insights into their structure. Following work analyzed optimization and approximation properties of infinite convex neural



(a) One-dimensional neuron (b) Two-dimensional neuron (c) Optimal breaklines in $\mathbb{R}^2$

Figure 1: An illustration of the geometric interpretation of optimal ReLU neurons. The breaklines/breakpoints pass through a subset of *special* training samples.

networks (Bach, 2017). Although these works advanced the understanding of convexity in infinite dimensional NNs, they also highlight the computational challenges inherent in training infinite dimensional models, including solving a finite dimensional non-convex problems to add a single neuron (Bach, 2017). On the other hand, it has been noted that the activation patterns of deep ReLU networks exhibit a structured yet poorly understood simplicity. The work Hanin and Rolnick (2019) highlighted that the actual number of activation regions a ReLU network learns in practice is significantly smaller than the theoretical maximum.

Previous studies (Fisher and Jerome, 1975; Mammen and Van De Geer, 1997) have investigated splines in relation to ℓ_1 extremal problems, demonstrating the adaptability of spline models to local data characteristics. More recent work (Balestriero et al., 2018) connected deep networks to spline theory, showing that many deep learning architectures could be interpreted as max-affine spline operators. Approximation properties of ReLU and squared ReLU networks with regularization were studied in Klusowski and Barron (2018). The authors in Savarese et al. (2019) considered infinitely wide univariate ReLU networks and showed that the minimum squared ℓ_2 norm fit is given by a linear spline interpolation. Another line of work (Parhi and Nowak, 2021; 2022) developed a variational framework to analyze functions learned by deep ReLU networks, revealing that these functions belong to a space similar to classical bounded variation-type spaces. In a similar spirit, connections to kernel Banach spaces via representer theorems were developed in Bartolucci et al. (2023). The work Unser (2019) introduced a general representer theorem that connects deep learning with splines and sparsity. In contrast, our results provide an independent and novel perspective on the optimal weights of a deep neural networks through geometric algebra, and may shed light into the spline theory of deep networks. In particular, the relation between linear splines and one-dimensional ReLU networks discovered in Savarese et al. (2019) is generalized to arbitrary dimension and depth in our work. The key ingredient in our analysis is the use of wedge products, which are not present in any work analyzing neural networks to the best of our knowledge.

The relationship between neural networks and geometric structures has been another area of research focus. Convex optimization and convex geometry viewpoint of neural networks has been extensively studied in recent work (Pilanci and Ergen, 2020; Ergen and Pilanci, 2021a; Bartan and Pilanci, 2021a; Ergen and Pilanci, 2021b;c; Wang and Pilanci, 2022; Wang et al., 2021; Ergen et al., 2022a; Lacotte and Pilanci, 2020; Bartan and Pilanci, 2021b). However, previous works have focused mostly on computational aspects of convex reformulations. This work provides an entirely new set of convex reformulations, which are different from the ones in the literature. Two main advantages of our approach are that our results hold for arbitrary depth and dimension, and that they provide a geometric interpretation and novel closed-form formulas, which can be used to polish any existing deep neural network.

1.2 Summary of results

The work in this paper diverges from other approaches by using Clifford's geometric algebra and convex analysis to characterize the structure of optimal neural network weights. We show that the optimal weights for deep ReLU neural networks can be found via the closed-form formula, $x_{j_1} \wedge \ldots \wedge x_{j_k}$, known as a k-blade in geometric algebra, with $\wedge$ signifying the wedge product. For each individual neuron, this expression involves a *special* subset of training samples indexed by $(j_1, \ldots, j_k)$, which may vary across neurons. Surprisingly,

the entire network training procedure can be reinterpreted as a purely discrete problem that identifies this unique subset for every neuron. Moreover, we show that this problem can be cast as a Lasso variable selection procedure over wedge product features, algebraically encoding the geometric structure of the training dataset.

The Unexpected Neuron Functionality

The conventional belief suggests that artificial neurons optimize their response by aligning with the relevant input samples (Carter et al., 2019), a notion inspired by the direction-sensitive neurons observed within the visual cortex (Hubel and Wiesel, 1968). However, this interpretation hits a dead end in large DNNs, with numerous neurons responding to unrelated features, making it nearly impossible to understand the specific role of individual neurons (O'Mahony et al., 2023). Contrary to this conventional wisdom, our findings reveal that ReLU neurons are, in fact, orthogonal to a specific set of data points, due to the properties of the wedge product. As a consequence, these neurons yield a null output for this distinct subset of training samples, diverging completely from the anticipated alignment. This outcome underscores a nuanced understanding; rather than merely aligning with input samples, the neurons assess the oriented distance relative to the affine hull encapsulated by the special subset of training samples. This concept is visually explained in Figure 1 (a-b), where ReLU activation in 1D and 2D is interpreted as an oriented distance function. The optimal breaklines of the ReLU neurons, i.e., $\{x : w^T x + b = 0\}$, intersect with a select group of *special* training samples, expressed using a simple equation involving wedge products, leading to a zero output from the neurons for these instances, as illustrated in Figure 1(c). This result challenges the traditional interpretations of the role of hidden layers in DNNs, and provides a fresh perspective on the inner workings of deep neural networks. We show for the first time that geometric algebra provides the right set of concepts and tools to work with such oriented distances, enabling the transformation of the problem into a simple convex formulation.

Decoding DNNs with Geometric Algebra

Our results show that within a deep neural network, when an input sample x is multiplied with a trained neuron, it yields the product $x^T \star (x_{j_1} \wedge \ldots \wedge x_{j_k})$. Leveraging geometric algebra, this product can be shown to be equal to the signed distance between x and the linear span of the point set $x_{j_1}, \ldots, x_{j_k}$, scaled by the length of the neuron. This allows the neurons to measure the oriented distance between the input sample and the affine hull of the special subset of samples (see Figure 1 for an illustration). Rectified Linear Unit (ReLU) activations transform negative distances, representing inverse orientations, to zero. When this operation is extended across a collection of neurons within a layer, the layer's output effectively translates the input sample into a coordinate system defined by the affine hulls of a special subset of training samples. Consequently, each layer is fundamentally tied solely to a specific subset of the training data. This subset can be identified by examining the weights of a trained network. Furthermore, with access to these training samples, the entirety of the network weights can be reconstructed using the wedge product formula. Moreover, when this operation is repeated through additional ReLU layers, our analysis reveals a geometric regularity within the space partitioning of the network, highlighting a consistent pattern of translations and interactions with dual vectors (see Figure 7). This geometric elucidation sheds fresh light on the mysterious roles played by the hidden layers in DNNs.

1.3 Notation

We use lower-case letters for vectors and upper-case letters for matrices. The notation $[n]$ represents integers from 1 to n. We use a multi-index notation to simplify indexing matrices and tensors. Specifically, $j = (j_1, ..., j_k)$ is a multi-index and $\sum_j$ denotes the summation operator over all indices included in the multi-index j, i.e., $\sum_{j_1} \cdots \sum_{j_k}$. Given a matrix $K \in \mathbb{R}^{n \times p}$, an ordinary index $i \in [n]$, and a multi-index $j = (j_1, ..., j_k)$ where $j_i \in [d_i]$ $\forall i \in [k]$, the notation K_{ij} denotes the $(i, v(j_1, ..., j_k))$-th entry of this matrix, where v represents the function that maps indices $(j_1, ..., j_k)$ to a one-dimensional index according to a column-major ordering. Formally, we can define $v(j_1, ..., f_d) := (j_1 - 1) + (j_2 - 1)d_1 + (j_3 - 1)d_1 d_2 + ... + (j_n - 1)d_1 d_2 ... d_{k-1}$. We allow the use of multi-index and ordinary indices together, e.g., $\sum_j a_j v_{j_1} \cdots v_{j_d} = \sum_{j_1, \cdots j_d} a_{(j_1, \cdots, j_d)} v_{j_1} \cdots v_{j_d}$. The notation $K_{i\cdot}$ denotes the i-th row of K. The notation $K_{\cdot j}$ denotes the j-th

column of K. The p-norm of a d-dimensional vector w for some $p \in (0, \infty)$ is represented by $\|w\|_p$, and it is defined as $\|w\|_p \triangleq (\sum_{i=1}^{d} |w_i|^p)^{1/p}$. In addition, we use the notation $\|w\|_0$ to denote the number of non-zero entries of w and $\|w\|_\infty$ to denote the maximum absolute value of the entries of w. We use the notation $\mathcal{B}_p^d$ for the unit p-norm ball in $\mathbb{R}^d$ given by $\{w \in \mathbb{R}^d : \|w\|_p \leq 1\}$. The notation $\mathbf{dist}(\mathbf{x}, \mathcal{Y})$ is used for the minimum Euclidean distance between a vector $\mathbf{x} \in \mathbb{R}^d$ and a subset $\mathcal{Y} \subseteq \mathbb{R}^d$. $\mathbf{Vol}(\mathcal{C})$ denotes d-volume of a subset $\mathcal{C} \subseteq \mathbb{R}^d$. The notation $(\cdot)_+$ is used the positive part of a real number. When applied to a scalar multiple of a pseudoscalar in a geometric algebra $\mathbb{G}^d$ such as $\alpha\mathbf{I} \in \mathbb{G}^d$, where $\mathbf{I}^2 = \pm 1$, the notation $(\alpha\mathbf{I})_+ = (\alpha)_+ \in \mathbb{R}$ represents the positive part of the scalar component of $\mathbf{I}$. We use the notation $\mathbf{Vol}_+(\cdot) = (\mathbf{Vol}(\cdot))_+$ to denote the positive part of signed volumes. We extend this notation to other functions, e.g., $\det_+(\cdot)$ denotes the positive part of the determinant, and $\mathbf{dist}_+(\cdot, \cdot)$ denotes the positive part of the Euclidean distance. $\mathbf{diam}(S)$ denotes the Euclidean diameter of a subset $S \subseteq \mathbb{R}^d$. $\mathbf{Span}(S)$ and $\mathbf{Aff}(S)$ denote the linear span and affine hull of a set of vectors S respectively. We overload scalar functions to apply to vectors and matrices element-wise. For instance $(Xw)_+$ denotes the ReLU activation applied to each entry of Xw. We use $\gtrsim$ to denote the inequality up to a constant factor.

2 Setting and Methodology

2.1 Preliminaries

Consider a deep neural network

$$f(x) = \sigma(W^{(L)} \cdots \sigma(W^{(1)} x + b^{(1)}) \cdots + b^{(L)}), \tag{1}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function, $W^{(1)}, \ldots, W^{(L)}$ are trainable weight matrices, $b^{(1)}, ..., b^{(L)}$ are trainable bias vectors and $x \in \mathbb{R}^d$ is the input. The activation function $\sigma(\cdot)$ operates on each element individually.

Training two-layer ReLU networks

We will begin by examining the regularized training objective for a two-layer neural network with ReLU activation function and m hidden neurons.

$$p^* \triangleq \min_{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}} \ell\Big(\sum_{j=1}^{m} \sigma(XW_j^{(1)} + 1_n b_j^{(1)}) W_j^{(2)} + b^{(1)}, y \Big) + \lambda \sum_{j=1}^{m} \|W_j^{(1)}\|_p^2 + \|W_j^{(2)}\|_p^2, \tag{2}$$

where $X \in \mathbb{R}^{n \times d}$ is the training data matrix, $W^{(1)} \in \mathbb{R}^{d \times m}$, $W^{(2)} \in \mathbb{R}^{m \times c}$ and $b^{(1)} \in \mathbb{R}^m$, $b^{(2)} \in \mathbb{R}$ are trainable weights, $\ell(\cdot, y)$ is a convex loss function, $y \in \mathbb{R}^n$ is a vector containing the training labels, and $\lambda > 0$ is the regularization parameter. Here we use the p-norm in the regularization of the weights. Initially, we will assume an output dimension of $c = 1$, and we will extend this to arbitrary values of c. Typical loss functions used in practice include squared loss, logistic loss, cross-entropy loss and hinge loss, which are convex functions.

When $p = 2$, the objective (2) reduces to the standard weight decay regularized NN problem

$$p^* = \min_{\theta,\, b} \ell\big(f_{\theta,b}(X), y\big) + \lambda \|\theta\|_2^2. \tag{3}$$

Here, θ is a vector containing the weights W_1 and W_2 in vectorized form and

$$f_{\theta,\, b}(X) \triangleq \sum_{j=1}^{m} \sigma(XW_j^{(1)} + 1_n b_j^{(1)}) W_j^{(2)} + 1_n b^{(2)}, \tag{4}$$

where 1_n is a vector of ones of length n. When b is set to zero, we refer to the neurons in the first layer as the *bias-free* neurons. When b is not set to zero, we refer to the neurons in the first layer as the *biased* neurons. Note that the bias terms are excluded from the regularization.

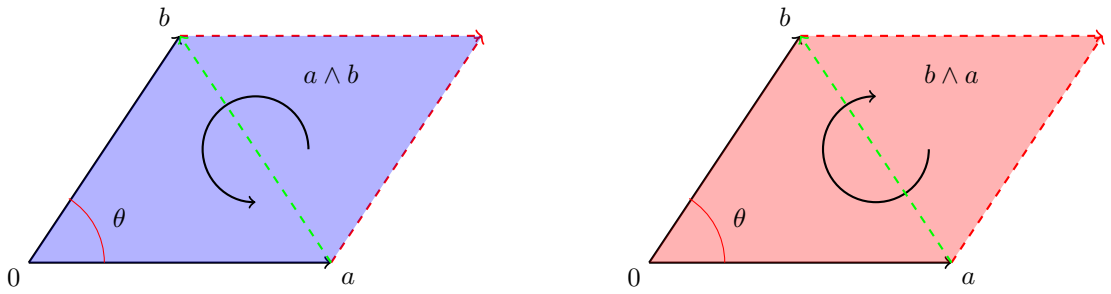


Figure 2: Wedge product of two-dimensional vectors.

Training deep ReLU networks

We will extend our analysis by also considering the objective in (3) using the deeper network model show in (1). It will be shown by a simple induction that the structural results for two-layer networks applies to blocks of layers in arbitrarily deep ReLU networks.

Augmented data matrix

In order to simplify our expressions for the case of $p = 1$, we augment the set of n training data vectors of dimension d by including d additional vectors from the standard basis of $\mathbb{R}^d$ and let $\tilde{n} = n + d$. Specifically, we define the augmented data samples as $\{x_i\}_{i=1}^{n+d} = \{x_i\}_{i=1}^{n} \cup \{e_i\}_{i=1}^{d}$, where $x_i \in \mathbb{R}^d$ represents the original training data points for $1 \leq i \leq n$ and e_i is the i-th standard basis vector in $\mathbb{R}^d$ for $i \geq n$. We define $\tilde{X} = [x_1, \cdots, x_{n+d}]^T$.

2.2 Geometric Algebra

Clifford's Geometric Algebra (GA) is a mathematical framework that extends the classical vector and linear algebra and provides a unified language for expressing geometric constructions and ideas (Artin, 2016). GA has found applications in classical and relativistic physics, quantum mechanics, electromagnetics, computer graphics, robotics and numerous other fields (Doran and Lasenby, 2003; Dorst et al., 2012). GA enables encoding geometric transformations in a form that is highly intuitive and convenient. More importantly, GA unifies several mathematical concepts, including complex numbers, quaternions, and tensors and provides a powerful toolset.

We consider GA over a d-dimensional Euclidean space, denoted as $\mathbb{G}^d$. The fundamental object in $\mathbb{G}^d$ is the *multivector*, $M = \langle M \rangle_0 + \langle M \rangle_1 + \ldots + \langle M \rangle_d$, which is a sum of vectors, bivectors, trivectors, and so forth. Here, $\langle M \rangle_k$ denotes the k-vector part of M. For instance, in the two-dimensional space $\mathbb{G}^2$, a multivector can be written as $M = a + v + B$, where a is a scalar, v is a vector, and B is a bivector. In this case, the basis elements are not just the canonical vectors e_1, e_2 but also the grade-2 element $e_1 e_2$.

A key operation in GA is the *geometric product*, denoted by juxtaposition of operands: ab. For vectors a and b, the geometric product can be expressed as $ab = a \cdot b + a \wedge b$, where $\cdot$ denotes the dot product and $\wedge$ denotes the wedge (or outer) product.

The dot product $a \cdot b$ is a scalar representing the projection of a onto b. The wedge product $a \wedge b$ is a bivector representing the oriented area spanned by a and b. In the geometric algebra $\mathbb{G}^d$, higher-grade entities (trivectors, 4-vectors, etc.) can be constructed by taking the wedge product of a vector with a bivector, a trivector, and so on. A k-blade is a k-vector that can be expressed as the wedge product of k vectors. For example, the bivector $a \wedge b$ is a 2-blade.

Figure 2 shows two important properties of the wedge product in $\mathbb{R}^2$:

1. Wedge product of two vectors represents the signed area of the paralleogram spanned by the two vectors. In the left figure, $a \wedge b$ is represented by the blue parallelogram. When a, b are two-dimensional vectors, the

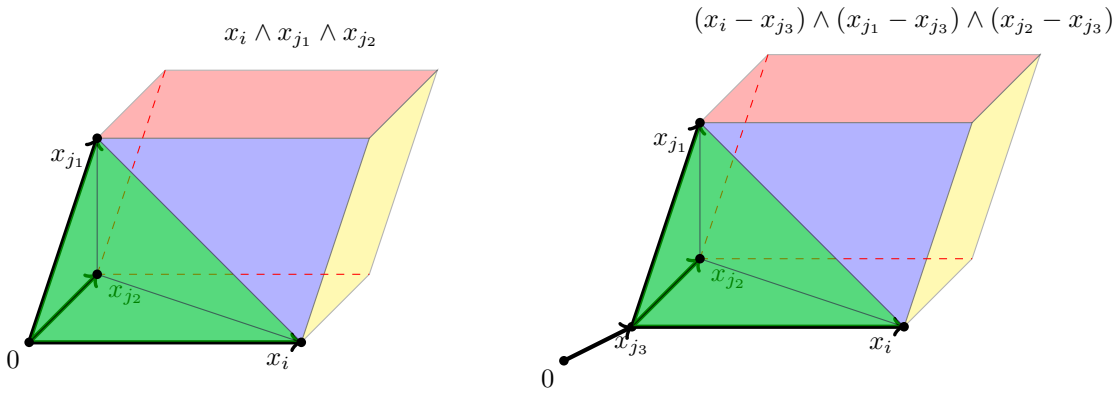


Figure 3: Wedge product representation of optimal neural networks in $\mathbb{R}^3$.

magnitude of $a \wedge b$ is equal to this area and is given by $\left| \det \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \right| = |a_1 b_2 - b_1 a_2|$. The sign of the area is determined by the orientation of the vectors: the area is positive when a can be rotated counter-clockwise to b and negative otherwise.

2. The wedge product is anti-commutative: In the right figure, $b \wedge a$ is represented by the red parallelogram. It has the same area as $a \wedge b = -b \wedge a$, but an opposite orientation. As a result, we have $a \wedge a = -a \wedge a = 0$.

By considering the half of the parallelogram, the area of the triangle in $\mathbb{R}^2$ formed by 0, a and b shown by the dashed line in Figure 2 is given by the magnitude of $\frac{1}{2} a \wedge b$. The signed area of a generic triangle in $\mathbb{R}^2$ formed by three arbitrary vectors a, b and c is given by

$$\frac{1}{2}(a - c) \wedge (b - c) = \frac{1}{2}\big(a \wedge b - a \wedge c - c \wedge b + c \wedge c\big) = \frac{1}{2}\big(a \wedge b + b \wedge c + c \wedge a\big)\,,$$

which is a consequence of the distributive property of the wedge product. Therefore, the signed area of the triangle is half the sum of the wedge products for each adjacent pair in the counter-clockwise sequence $a \to b \to c \to a$ encircling the triangle. In higher dimensions, the scalar part of the wedge product represents signed the volume of a parallelotope spanned by the vectors (see Figure 3).

The metric signature in $\mathbb{G}^d$ over the Euclidean space is characterized by all positive signs, indicating that all unit basis vectors are mutually orthogonal and have unit Euclidean norm. Let us call the standard (Kronecker) basis in d-dimensional Euclidean space $e_1, \ldots, e_d$, which satisfies $e_i^2 = 1 \, \forall i$ and $e_i e_j = -e_j e_i \, \forall i \neq j$. The wedge product $\mathbf{I} \triangleq e_1 \wedge \cdots \wedge e_d = e_1 \ldots e_d$ represents the highest grade element and is defined to be the unit pseudoscalar. The inverse of $\mathbf{I}$ is defined as $\mathbf{I}^{-1} = e_d \ldots e_1$, and satisfies $\mathbf{I}^{-1}\mathbf{I} = 1$. Squaring $\mathbf{I}$, we obtain $\mathbf{I}^2 = (e_1 e_2)^2 = e_1 e_2 e_1 e_2 = -1$, analogous to the unit imaginary scalar in complex numbers, which is a subalgebra of $\mathbb{G}^2$.

The ℓ_2 norm of a multivector is defined as the square root of the sum of the squares of the scalar coefficients of its basis k-vectors. For a multivector M it holds that $\|M\|_2^2 = \langle M^\dagger M \rangle_0$, where $M^\dagger$ is the reversion of M. $M^\dagger$ is analogous to complex conjugation and is defined by three properties: (i) $(MN)^\dagger = N^\dagger M^\dagger$, (ii) $(M + N)^\dagger = M^\dagger + N^\dagger$, (iii) $M^\dagger = M$ when M is a vector. For instance, $(e_1 e_2 e_3 e_4)^\dagger = e_4 e_3 e_2 e_1$ and $(e_1 + e_2 e_3)^\dagger = e_1 + e_3 e_2$. We define the ℓ_1 norm of a multivector as the sum of the absolute values of each component. The definition of inner and wedge products can be naturally extended to multivectors. In particular, the inner-product between two k-vectors $M = \alpha_1 \wedge \cdots \wedge \alpha_k$ and $N = \beta_1 \wedge \cdots \beta_k$ is defined by the Gram determinant $M \cdot N \triangleq \det(\langle \alpha_i, \beta_j \rangle_{i,j=1}^k)$.

Hodge dual

A k-blade can be viewed as a k-dimensional oriented parallelogram. For each such paralleogram, we may associate a $(d - k)$-dimensional orthogonal complement. This duality between k-vectors and $(d - k)$-vectors

is established through the Hodge star operator $\star$. For every pair of k-vectors $M, N \in \mathbb{G}^d$, there exists a unique $(d-k)$-vector $\star M \in \mathbb{G}^d$ with the property that

$$\star M \wedge N = (M \cdot N)\, e_1 \wedge \cdots \wedge e_d = (M \cdot N)\, \mathbf{I}.$$

We may also express the Hodge dual of a k-vector M as $\star M = M\mathbf{I}^{-1}$, where $\mathbf{I}^{-1}$ is the inverse of the unit pseudoscalar. This linear transformation from d-vectors to $d-k$ vectors defined by $M \to \star M$ is the Hodge star operator. An example in $\mathbb{G}^3$ is $\star e_1 = e_1\mathbf{I}^{-1} = e_1 e_3 e_2 e_1 = e_3 e_2$.

Generalized cross product and wedge products

The usual cross product of two vectors is only defined in $\mathbb{R}^3$. However, the wedge product can be used to define a generalized cross product in any dimension. The generalized cross product in higher dimensions is an operation that takes in $d-1$ vectors in an $\mathbb{R}^d$ and outputs a vector that is orthogonal to all of these vectors. The generalized cross product of the vectors $v_1, v_2, \ldots, v_{n-1}$ can be defined via the Hodge dual of their wedge product as $\times(v_1, v_2, \ldots, v_{n-1}) \triangleq \star(v_1 \wedge v_2 \wedge \ldots \wedge v_{n-1})$.. It holds that $\times(v_1, v_2, \ldots, v_{n-1}) \cdot v_i = 0$ for all $i = 1, \ldots, n-1$. The signed distance of a vector x to the linear span of a collection of vectors $x_1, ..., x_{d-1}$ can be expressed via the generalized cross product and wedge products as

$$\mathbf{dist}(x, \mathbf{Span}(x_1, \ldots, x_{d-1})) = \frac{\times(x_1, \ldots, x_{d-1})^T x}{\| \times (x_1, \ldots, x_{d-1})\|_2} = \frac{\star(x \wedge x_1 \wedge \cdots \wedge x_{d-1})}{\|x_1 \wedge \cdots \wedge x_{d-1}\|_2}.$$

This formulation stems from an intuitive geometric principle: the ratio of the volume of a parallelotope $\mathcal{P}(x, x_1, \ldots, x_{d-1})$, which is spanned by the vectors $x, x_1, \ldots, x_{d-1}$, to the volume of its base $\mathcal{P}(x_1, \ldots, x_{d-1})$, the parallelotope formed by $x_1, \ldots, x_{d-1}$ alone. This ratio effectively captures the height of the parallelotope relative to its base, which corresponds to the distance of x from the subspace spanned by $x_1, \ldots, x_{d-1}$. Note that the Hodge dual $\star$ transforms the d-vector in the numerator into a scalar. We provide a subset of other important properties of the generalized cross product in Section 6.1 of the Appendix.

2.3 Convex duality

Convexity and duality plays a key role in the analysis of optimization problems (Boyd and Vandenberghe, 2004). Here, we show how the convex dual of a non-convex neural network training problem can be used to analyze optimal weights.

Convex duals of neural network problems

The non-convex optimization problem (2) has a convex dual formulation derived in recent work (Pilanci and Ergen, 2020; Ergen and Pilanci, 2021a) given by

$$p^* \geq d^* \triangleq \max_{v \in \mathbb{R}^n} \; -\ell^*(v, y) \quad \text{s.t.} \quad |v^T \sigma(Xw)| \leq \lambda, \, \forall w \in \mathcal{B}_p^d, \tag{5}$$

where we take the network to be bias-free (see Section 8.2 for biased neurons). Here, $\ell^*(\cdot, y)$ is the convex conjugate of the loss function $\ell(\cdot, y)$ defined as $\ell^*(v, y) \triangleq \sup_{q \in \mathbb{R}^n} v^T q - \ell(q, y)$ and $\mathcal{B}_p^d$ is the unit p-norm ball in $\mathbb{R}^d$. Moreover, it was shown in Pilanci and Ergen (2020) that when σ is the ReLU activation, strong duality holds, i.e., $p^* = d^*$, when the number of neurons m exceeds a critical threshold. The value of this threshold can be determined from an optimal solution of the dual problem in (5). This result was extended to deeper ReLU networks in Ergen and Pilanci (2021c) and to deep threshold activation networks in Ergen et al. (2022b).

Our main strategy to analyze the optimal weights is based on analyzing the extreme points of the dual constraint set in (5). In the following section, we present our main results.

3 Theoretical Results

3.1 One-dimensional data

We start with the simplest case where the training data is one-dimensional, i.e., $d = 1$ and the number of training points, n is arbitrary.

Theorem 1. *For all values of the regularization norm $p \in [1, \infty)$, the two-layer neural network problem in (2) can be recast as the following ℓ_1-regularized convex optimization problem*

$$\min_{\substack{z \in \mathbb{R}^{2n} \\ t \in \mathbb{R}}} \ell\big(Kz + 1_n t, y\big) + \lambda \|z\|_1 \,, \tag{6}$$

where the entries of the matrix K are given by

$$K_{ij} \triangleq \begin{cases} (x_i - x_j)_+ & 1 \leq j \leq n \\ (x_{j-n} - x_i)_+ & n < j \leq 2n, \end{cases} \tag{7}$$

and the number of neurons obey $m \geq \|z^*\|_0$. An optimal network can be constructed as

$$f(x) = \sum_{j=1}^{n} z_j^* (x - x_j)_+ + \sum_{j=1}^{n} z_{j+n}^* (x_j - x)_+ + t^*, \tag{8}$$

where z^* and t^* are optimizers of (6).

Remark. It will be revealed in the following sections that the term $(x_i - x_j)_+$, as present in (7), stands for the wedge product $\left(\begin{bmatrix} x_i \\ 1 \end{bmatrix} \wedge \begin{bmatrix} x_j \\ 1 \end{bmatrix} \right)_+$, yielding the positive part of the signed length of the interval $[x_i, x_j]$. Appending 1 to the vectors is due to the presence of bias in the neurons. This quantity can also be seen as a directional distance we denote as $\mathbf{dist}_+(x_i, x_j)$, which will be generalized to higher dimensions in the sequel. As we delve into higher dimensional NN problems, the above wedge product expression will be substituted by the positive part of the signed volume of higher dimensional simplices such as triangles and parallelograms.

Remark. This result is a refinement of the linear spline characterization of one-dimensional infinitely wide ReLU NNs (Savarese et al., 2019; Parhi and Nowak, 2020). The linear spline dictionary is given by the collection of ramp functions $\{(x - x_j)_+, (x_j - x)_+\}_{j=1}^n$, which is well-known in adaptive regression splines (Friedman, 1991). Our work is the first to recognize this dictionary via wedge products, characterize it as a finite dimensional Lasso problem, and associate it to volume forms. This enables us to generalize the result to higher dimensions and arbitrarily deep ReLU networks. It is important to note that unlike the existing literature on infinite neural networks (Bach, 2017; Bengio et al., 2005), our characterization of the dictionaries is discrete rather than continuous, enabling standard convex programming.

An important feature of the optimal network in (8) is that the break points are located only at the training data points. In other words, the prediction $f(x)$ is a piecewise linear function whose slope only may change at the training points with at most n breakpoints. However, since the optimal z^* is sparse, the number of pieces is at most $\|z^*\|_0$, and can be smaller than n.

We note that convex programs for deep networks trained for one-dimensional data were considered in Ergen et al. (2022b); Zeger et al. (2024). In Ergen and Pilanci (2021a), it was shown that the optimal solution to (3) may not be unique and may contain break points at locations other than the training data points. However, Theorem 1 reveals that at least one optimal solution is in the form of (8). In addition, it is shown that the solution is unique when the bias terms are regularized (Boursier and Flammarion, 2023) and a skip connection is included. We discuss uniqueness in Section 5.

3.2 Two-dimensional data

We now consider the case $d = 2$ and use the two-dimensional geometric algebra $\mathbb{G}^2$. Interestingly, we will observe that the volume of the interval $[x_i, x_j]$ appearing above generalizes to the volume of a triangle. We

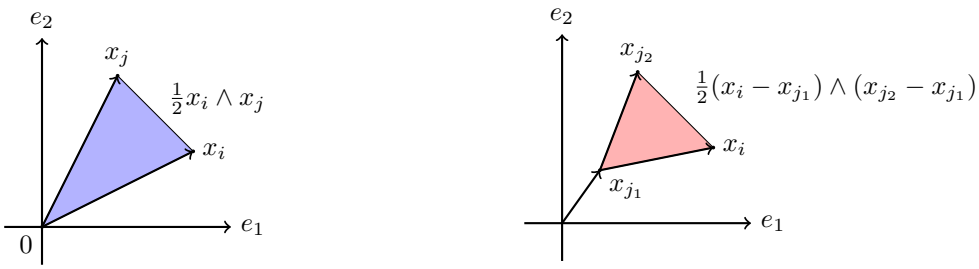


Figure 4: Illustration of the matrix K for neural networks without bias (left) and with bias (right) via the triangular area defined in the convex program from Theorem 2 in $\mathbb{R}^2$.

will observe that the ℓ_1 and ℓ_2-regularized problems exhibit certain differences from one another. We start with the ℓ_1-regularized problem $p = 1$, since the form of the convex program is simpler to state due to the polyhedral nature of the ℓ_1 norm. In the case of $p = 2$, we require a mild regularity condition on the dataset to handle the curvature of the ℓ_2 norm.

3.2.1 ℓ_1 regularization - neurons without biases

Theorem 2. *For $p = 1$ and $d = 2$, the two-layer neural network problem without biases can be recast as the following convex ℓ_1-regularized optimization problem*

$$\min_{z \in \mathbb{R}^{\tilde{n}}} \ell\big(Kz, y\big) + \lambda \|z\|_1. \tag{9}$$

Here, the matrix $K \in \mathbb{R}^{\tilde{n} \times \tilde{n}}$ is defined as $K_{ij} = \kappa(x_i, x_j)$ where

$$\kappa(x, x') := \frac{(x \wedge x')_+}{\|x'\|_1} = \frac{2\mathbf{Vol}_+(\triangle(0, x, x'))}{\|x'\|_1}, \tag{10}$$

provided that the number of neurons satisfy $m \geq \|z^*\|_0$. Here, $\triangle(0, x_i, x_j)$ denotes the triangle formed by the path $0 \to x_i \to x_j$, and $\mathbf{Vol}_+(\cdot)$ denotes the positive part of the signed area of this triangle.

An optimal network can be constructed as follows:

$$f(x) = \sum_{j=1}^{\tilde{n}} z_j^* \kappa(x, x_j),$$

where z^* is an optimal solution to (9). The optimal first layer neurons are given by a scalar multiple of the generalized cross product, $\times x_j = \star x_j$, with breaklines $x \wedge x_j = 0$, corresponding to non-zero z_j^* for $j \in [\tilde{n}]$.

Remark. The optimal hidden neurons given by the generalized cross products, $\times x_j$, are orthogonal to the training data points x_j for $j = 1, ..., \tilde{n}$. Therefore, the breaklines of each ReLU neuron pass through the origin and some data point x_j. Note that $(x \wedge x_j)_+ = (x_1 x_{j2} - x_2 x_{j1})_+$ is a ReLU ridge function.

We provide an illustration the matrix K in Figure 4. The signed area of the triangle formed by the path $0 \to x_i \to x_j$, denoted by the wedge product $\frac{1}{2} x_i \wedge x_j$ is positive when this path is ordered counterclockwise and negative otherwise. In this figure, the positive part of this signed area given by $\mathbf{Vol}_+(\triangle(0, x_i, x_j)) = \frac{1}{2}(x_i \wedge x_j)_+$ is non-zero only when x_i is to the right of the line passing through the origin and x_j. When bias terms are added to the neurons, the matrix K changes as shown in the right panel of Figure 4 as shown in Theorem 3 (see also Section 8.2 in the Supplementary Material).

3.2.2 ℓ_2 regularization (weight decay) - neurons with biases

Now we show that similar results holds for the ℓ_2 regularization (weight decay) for a near-optimal solution, also demonstrate the case with biases. We first quantify near-optimality as follows:

Definition 1 (Near-optimal solutions)**.** We call a set of parameters that achieve the cost $\hat{p}$ in the two-layer NN objective (3) ϵ-optimal if

$$p^* \leq \hat{p} \leq (1+\epsilon)p^*\,, \tag{11}$$

where p^* is the global optimal value of (3) and $\epsilon > 0$.

Definition 2 (Range dispersion in $\mathbb{R}^2$)**.** We call that a two-dimensional dataset is ϵ-dispersed for some $\epsilon \in (0, \frac{1}{2}]$ if

$$|\theta_{i+1} - \theta_i| \pmod{\pi} \leq \epsilon\pi \quad \forall i \in [n], \tag{12}$$

where θ_i are the angles of the vector x_i with respect to the horizontal axis, i.e.,

$$x_i = \|x_i\|_2 \big[\cos(\theta_i) \;\; \sin(\theta_i)\big]^T$$

sorted in increasing order. We call the dataset locally ϵ-dispersed if the above condition holds for the set of differences $\{x_i - x_j\}_{i=1}^n$ for all $j \in [n]$.

Range dispersion measures the diversity of the normal planes corresponding to the training data. Local dispersion holds when the data centered at any training sample is dispersed. In the left panel of Figure 5, we illustrate the range dispersion condition for a two-dimensional dataset.

Theorem 3. *Suppose that the training set $\{x_1, \ldots, x_n\}$ is locally ϵ-dispersed. For $p = 2$ and $d = 2$, an ϵ-optimal network can be found via the following convex optimization problem*

$$\min_{\substack{z \in \mathbb{R}^{\binom{\tilde{n}}{2}} \\ t \in \mathbb{R}}} \ell\big(Kz + \mathbf{1}t, y\big) + \lambda\|z\|_1, \tag{13}$$

when the number of neurons obey $m \geq \|z^*\|_0$. Here, the matrix $K \in \mathbb{R}^{\tilde{n} \times \binom{\tilde{n}}{2}}$ is defined as $K_{ij} := \kappa(x_i, x_{j_1}, x_{j_2})$ for $j = (j_1, j_2)$, where

$$\kappa(x, x', x'') = \frac{\big(x \wedge x' + x' \wedge x'' + x'' \wedge x\big)_+}{\|x' \wedge x''\|_2} = \frac{2\mathbf{Vol}_+(\triangle(x, x', x''))}{\|x' - x''\|_2} = \mathbf{dist}_+(x, \mathbf{Aff}(x', x'')),$$

for $j = (j_1, j_2)$. The ϵ-optimal neural network can be constructed as

$$f(x) = \sum_{j=(j_1,j_2)} z_j^* \kappa(x, x_{j_1}, x_{j_2}),$$

where z^* is an optimal solution to (13). The optimal hidden neurons and biases are given by a scalar multiple of $\star(x_{j_1} - x_{j_2})$, and $-\star(x_{j_2} \wedge (x_{j_1} - x_{j_2}))$ respectively, with breaklines $(x - x_{j_2}) \wedge (x_{j_1} - x_{j_2}) = 0$ for $j = (j_1, j_2)$ corresponding to non-zero z_j^*.

Remark. We note that the form of the near-optimal NN for $p = 2$ is near identical to the $p = 1$ case, for which the result is exact. This discrepancy is due to the polyhedral nature of the dual problem with ℓ_1 regularization (see Figure 40 of Appendix II). In Section 4, we present numerical evidence that the decision regions of optimal NNs with $p = 1$ and $p = 2$ are near identical for small values of λ.

3.3 Arbitrary dimensions

Now we consider the generic case where d and n are arbitrary and we use the d-dimensional geometric algebra $\mathbb{G}^d$. Suppose that $X \in \mathbb{R}^{n \times d}$ is a training data matrix such that $\mathbf{rank}(X) = d$ without loss of generality. Otherwise, we can reduce the dimension of the problem to $\mathbf{rank}(X)$ using Singular Value Decomposition (see Lemma 23 in the Supplementary Material), hence d can be regarded as the rank of the data. Since many datasets encountered in machine learning problems are close to low rank, this method can be used to reduce the number of variables in the convex programs we will introduce in this section.

3.3.1 ℓ_1 regularization - neurons without biases

Theorem 4. *The 2-layer neural network problem in (2) when $p = 1$ and biases set to zero is equivalent to the following convex Lasso problem*

$$\min_z \ell\big(Kz, y\big) + \lambda \|z\|_1, \tag{14}$$

provided that the number of neurons satisfy $m \geq \|z^*\|_0$. The matrix K is defined as $K_{ij} = \kappa(x_i, x_{j_1}, ..., x_{j_{d-1}})$ for $j = (j_1, ..., j_{d-1})$, where

$$\kappa(x, u_1, ..., u_{d-1}) = \frac{x^T \times (x_{j_1}, \cdots, x_{j_{d-1}})}{\| \times (x_{j_1}, \cdots, x_{j_{d-1}})\|_2} = \frac{\mathbf{Vol}_+(\mathcal{P}(x,\, u_1,\, ...,\, u_{d-1}))}{\| \star (u_1 \wedge \, ... \, \wedge u_{d-1})\|_1}, \tag{15}$$

the multi-index $j = (j_1, ..., j_{d-1})$ indexes over all combinations of $d-1$ rows x_{j_1}, ..., $x_{j_{d-1}} \in \mathbb{R}^d$ of $\tilde{X} \in \mathbb{R}^{\tilde{n} \times d}$. An optimal neural network can be constructed as follows:

$$f(x) = \sum_{j=(j_1,\ldots,j_{d-1})} z_j^* \kappa(x, x_{j_1}, ..., x_{j_{d-1}}),$$

where z^* is an optimal solution to (14). The optimal neurons are given by a scalar multiple of the generalized cross-product $\times(x_{j_1}, \cdots, x_{j_{d-1}}) = \star(x_{j_1} \wedge \cdots \wedge x_{j_{d-1}})$, with breaklines $x \wedge x_{j_1} \wedge \cdots \wedge x_{j_{d-1}} = 0$, corresponding to non-zero z_j^* for $j = (j_1, \cdots, j_{d-1})$.

We recall that $\mathcal{P}(x,\, u_1,\, ...,\, u_{d-1})$ denotes the parallelotope formed by the vectors x, u_1, ..., u_{d-1}, and the positive part of the signed volume of this parallelotope is given by $\mathbf{Vol}_+(\mathcal{P}(x,\, u_1,\, ...,\, u_{d-1}))$.

Remark. The optimal neurons are orthogonal to $d-1$ data points, i.e., $\times(x_{j_1}, \cdots, x_{j_{d-1}}) \cdot x_i = 0$ for all $i \in \{j_1, \cdots, j_{d-1}\}$. Therefore, the hidden ReLU neuron is activated on a halfspace defined by the hyperplane that passes through data points $x_{j_1}, \cdots, x_{j_{d-1}}$.

The proof of this theorem can be found in Section 9.2 of the Supplementary Material.

Remark. We note that the combinations can be taken over $d-1$ linearly independent rows of X since otherwise the volume is zero and corresponding weights can be set to zero. Moreover, the permutations of the indices x_{j_1}, ..., $x_{j_{d-1}}$ may only change the sign of the volume $\mathbf{Vol}(\mathcal{P}(x_i,\, x_{j_1},\, ...,\, x_{j_{d-1}}))$. Therefore, it is sufficient to consider each subset that contain $d-1$ linearly independent data points and compute $\mathbf{Vol}_+(\pm\mathcal{P}(x_i,\, x_{j_1},\, ...,\, x_{j_{d-1}}))$ for each subset. The cost of enumerating over all size d subsets is $O(\binom{n}{d})$.

3.3.2 ℓ_2 regularization - neurons with biases

We begin by defining a parameter that sets an upper limit on the diameter of the chambers in the arrangement generated by the rows of the training data matrix X.

Definition 3. We define the Maximum Chamber Diameter, denoted as $\mathscr{D}(X)$, using the following equation:

$$\mathscr{D}(X) := \max_{\substack{w,\, v \in \mathbb{R}^d,\ \|w\|_2 = \|v\|_2 = 1 \\ \operatorname{sign}(Xw) = \operatorname{sign}(Xv)}} \|w - v\|_2. \tag{16}$$

Here w and v are unit-norm vectors in $\mathbb{R}^d$, such that the sign of the inner-product with the data rows are the same.

We call a dataset ϵ-dispersed if $\mathscr{D}(X) \leq \epsilon$, and locally ϵ-dispersed when the dataset centered at any training sample is ϵ dispersed, i.e., $\mathscr{D}(X - 1x_j^T) \leq \epsilon \ \forall j \in [n]$.

Remark. The quantity $\mathscr{D}(X)$ is a generalization of the 2D range dispersion in Definition 2 to arbitrary dimensions, and captures the diversity of the ranges of the hyperplanes whose normals are training points $\{x_i\}_{i=1}^n$. We prove in Section 3.5 that when the data is randomly generated, e.g., i.i.d. from a Gaussian distribution, the maximum chamber diameter $\mathscr{D}(X)$ is bounded by $(\frac{d}{n})^{1/4}$ with probability that approaches 1 exponentially fast. In Lemma 8 of Section 3.5, we show that this implies Gaussian data is ϵ-dispersed and locally ϵ-dispersed when as $n \gtrsim \epsilon^{-4} d$ with high probability.

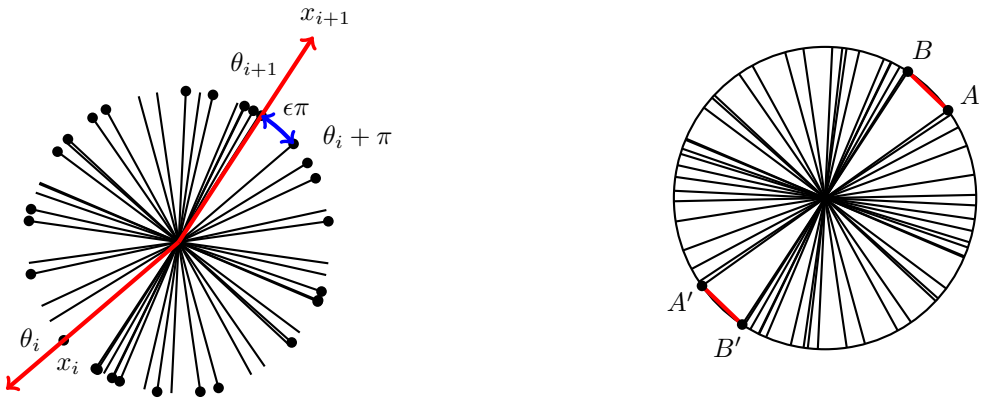


Figure 5: (left) An illustration of the angular dispersion condition. The angle between the span of two consecutive vectors x_i and x_{i+1} is bounded by $\epsilon\pi$. (right) The maximum chamber diameter of this line arrangement is the Euclidean distance between A and B, i.e., $\mathscr{D}(X) = \|A - B\|$.

Theorem 5. *Consider the following convex optimization problem*

$$\hat{p}_\lambda := \min_z \ell\big(Kz, y\big) + \lambda\|z\|_1\,. \tag{17}$$

The matrix K is defined as $K_{ij} = \kappa(x_i, x_{j_1}, ..., x_{j_{d-1}})$ for bias-free and $K_{ij} = \kappa_b(x_i, x_{j_1}, ..., x_{j_{d-1}})$ for biased neurons where

$$\kappa(x, u_1, ..., u_{d-1}) = \frac{\big(x \wedge u_1 \wedge \cdots \wedge u_{d-1}\big)_+}{\|u_1 \wedge \ ... \ \wedge u_{d-1}\|_2} = \mathbf{dist}_+\big(x,\, \mathbf{Span}(u_1,\, ...,\, u_{d-1})\big)$$
$$\kappa_b(x, u_1, ..., u_d) = \frac{\big((x - u_d) \wedge (u_1 - u_d) \wedge \cdots \wedge (u_{d-1} - u_d)\big)_+}{\|(u_1 - u_d) \wedge \ ... \ \wedge (u_{d-1} - u_d)\|_2} = \mathbf{dist}_+\big(x,\, \mathbf{Aff}(u_1,\, ...,\, d)\big)$$

and the multi-index $j = (j_1, ..., j_{d-1})$ is indexing over all combinations of $d-1$ rows $x_{j_1}, ..., x_{j_{d-1}} \in \mathbb{R}^d$ of $X \in \mathbb{R}^{n\times d}$. When the maximum chamber diameter satisfies $\mathscr{D}(X) \leq \epsilon$ for bias-free neurons or $\mathscr{D}(X - 1x_j^T) \leq \epsilon\, \forall j \in [n]$ for biased neurons, for some $\epsilon \in (0, 1)$, we have the following approximation bounds

$$p^* \leq \hat{p}_\lambda \leq \frac{1}{1-\epsilon} p^*, \tag{18}$$
$$\hat{p}_{(1-\epsilon)\lambda} \leq p^* \leq \hat{p}_\lambda \leq p^* + \frac{\epsilon}{1-\epsilon}\lambda R^*, \tag{19}$$

Here, p^* is the value of the optimal NN objective in (2) (with or without bias terms) and R^* is the corresponding weight decay regularization term of an optimal NN. Bias-free and biased networks that achieves the cost $\hat{p}_\lambda$ in (2) are

$$f(x) = \sum_{j=(j_1,...,j_{d-1})} z_j^* \kappa(x_{j_1},\, ...,\, x_{j_{d-1}}) \quad \textit{and} \quad f(x) = \sum_{j=(j_1,...,j_{d-1})} z_j^* \kappa_b(x_{j_1},\, ...,\, x_{j_{d-1}})\,,$$

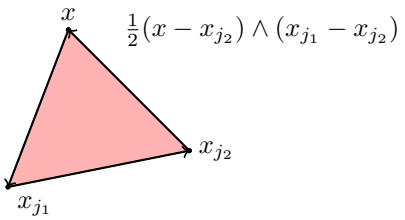
respectively, where z^* is an optimal solution to (17).

Remark. The above result shows that the convex optimization produces a near-optimal solution when the chamber diameter is small, which is expected when the number of data points is large.

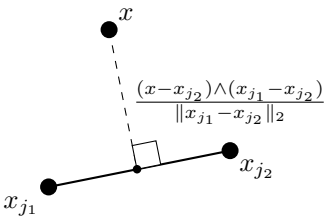
3.4 Illustrative Calculations in Geometric Algebra

Here we illustrate applications of the closed-form geometric algebra expressions in $\mathbb{R}^2$. Consider a dataset with two training samples of dimension two given by

$$x_1 = \left[\begin{smallmatrix}1\\1\end{smallmatrix}\right] = e_1 + e_2 \qquad \text{and} \qquad x_2 = \left[\begin{smallmatrix}0\\2\end{smallmatrix}\right] = 2e_2.$$



(a) The area of triangle formed by the points x, x_{j_1} and x_{j_2} is given by a wedge product.



(b) Distance to affine hull is given by a wedge product normalized by Euclidean distance.

Figure 6: Wedge product representation of distance to affine hull (a) and triangular area (b).

Theorem 4 for $p = 1$ implies that an optimal neuron corresponding to a sample x is given by the dual $\star x := x\mathbf{I}^{-1}$. We recall that $\mathbf{I} = e_1 e_2 \dots e_d = e_1 e_2$ when $d = 2$, and $\mathbf{I}^{-1} = e_2 e_1$. Hence, we have $\star x = x e_2 e_1$ in $\mathbb{G}^2$. Next, recall that for $p = 1$, we augment the training set with two standard basis vectors of $\mathbb{R}^2$ and obtain $\{\tilde{x}_i\}_{i=1}^4 = \{e_1 - e_2, 2e_1, e_1, e_2\}$. For each training sample, we calculate these neurons as follows

$$w_1 = \star\tilde{x}_1 = (e_1 + e_2)e_2 e_1 = \underbrace{e_1 e_2}_{-e_2 e_1} e_1 + \underbrace{e_2 e_2}_{1} e_1 = -e_2 \underbrace{e_1 e_1}_{1} + e_1 = -e_2 + e_1$$

and

$$w_2 = \star\tilde{x}_2 = 2e_2 e_2 e_1 = 2e_1.$$

We find two additional neurons from the augmented training data points e_1 and e_2 as $w_3 = e_2$ and $w_4 = -e_1$ respectively using the same process. This yields $m = 4$ neurons, which are orthogonal to $\{\tilde{x}_i\}_{i=1}^4$.

Theorem 2 provides the optimal ReLU network with no biases as

$$f(x) = \sum_j (x^T w_j)_+ \alpha_j = \sum_{j=1}^4 \frac{\mathbf{Vol}(\triangle(0, x, \tilde{x}_j))_+}{\|\tilde{x}_j\|_1} \alpha_j,$$

where $\{\alpha_j\}_{j=1}^m$ are determined via the Lasso problem $\arg\min_\alpha \sum_{i=1}^n (f(x_i) - y_i)^2 + \lambda\|\alpha\|_1$ as described earlier in (9).

Let us illustrate some variants of this network which has no bias terms and trained with p-regularization where $p = 1$. For the optimal network with bias terms, we replace the volume terms with $\mathbf{Vol}(\triangle(x, \tilde{x}_{j_1}, \tilde{x}_{j_2}))$. This volume term has the loop based expansion

$$\begin{aligned} 2\mathbf{Vol}(\triangle(x, \tilde{x}_{j_1}, \tilde{x}_{j_2})) = (x - \tilde{x}_{j_2}) \wedge (\tilde{x}_{j_1} - \tilde{x}_{j_2}) &= x \wedge \tilde{x}_{j_1} - x \wedge \tilde{x}_{j_2} - \tilde{x}_{j_2} \wedge \tilde{x}_{j_1} - \underbrace{\tilde{x}_{j_2} \wedge \tilde{x}_{j_2}}_{0} \\ &= x \wedge \tilde{x}_{j_1} + \tilde{x}_{j_1} \wedge \tilde{x}_{j_2} + \tilde{x}_{j_2} \wedge x. \end{aligned}$$

In other words, the volume of the triangle formed by vertices $x, \tilde{x}_{j_1}, \tilde{x}_{j_2}$ can be expanded as the wedge product of consecutive pairs of vectors taken over the loop $x \to \tilde{x}_{j_1} \to \tilde{x}_{j_2} \to x$. This is illustrated in Figure 6a. Let us now illustrate the role of regularization. For the p-regularized network with $p = 2$, Theorem 3 implies that we replace the volume terms with

$$\mathbf{dist}(x, \mathbf{Aff}(x_{j_1}, x_{j_2})) = \frac{(x - x_{j_2}) \wedge (x_{j_1} - x_{j_2}))}{\|x_{j_1} - x_{j_2}\|_2} = \frac{\mathbf{Vol}(\triangle(x, x_{j_1}, x_{j_2}))}{\mathbf{dist}(x_{j_1}, x_{j_2})}.$$

This is illustrated in Figure 6b. The change from the volume to the distance to affine hull is a consequence of the division by $\|x_{j_1} - x_{j_2}\|_2$, which is the length of the base (the line segment $\mathbf{Aff}(x_{j_1}, x_{j_2})$) of the triangle $\triangle(x, x_{j_1}, x_{j_2})$.

3.5 Data isometry and chamber diameter

Results from high dimensional probability can be used to establish bounds on the chamber diameter of a hyperplane arrangement generated by a random collection of training points. A similar analysis was considered in Plan and Vershynin (2014) for the purpose of dimension reduction. We first show that the chamber diameter is small when the training dataset satisfies an isometry condition.

Lemma 6 (Isometry implies small chamber diameter)**.** *Suppose that the following condition holds*

$$(1-\epsilon)\|w\|_2 \leq \frac{1}{\alpha n}\|Xw\|_1 \leq (1+\epsilon)\|w\|_2, \quad \forall w \in \mathbb{R}^d, \tag{20}$$

where $\alpha \in \mathbb{R}$ is fixed. Then, the chamber diameter $\mathscr{D}(X)$ defined in (16) is bounded by $4\sqrt{\epsilon}$.

Next, we show that the isometry condition is satisfied when the training dataset is generated from a random distribution. Consequently, we obtain a bound on the chamber diameter of the hyperplane arrangement generated by the random dataset.

Lemma 7 (Random datasets have small chamber diameter)**.** *Suppose that $x_1, \cdots, x_n \sim \mathcal{N}(0, I_d)$ are n random vectors sampled from the standard d-dimensional multivariate normal distribution and let $X = [x_1, \cdots, x_n]^T$. Then, the ℓ_2 diameter $\mathscr{D}(X)$ satisfies*

$$\mathscr{D}(X) \leq 9\Big(\frac{d}{n}\Big)^{1/4}, \tag{21}$$

with probability at least $1 - 2e^{-d/2}$.

Remark. We note that the constant 9 in the above lemma can be improved to 1 by using the more refined analysis due to Gordon (Gordon, 1985). Moreover, the result can be extended to sub-Gaussian data distributions. However, we use Lemma 6 since it is simpler and suffices for our purposes.

Lemma 8 (Random datasets are dispersed and locally dispersed)**.** *Suppose that $x_1, \cdots, x_n \sim \mathcal{N}(0, I_d)$ are n random vectors sampled from the standard d-dimensional multivariate normal distribution and let $X = [x_1, \cdots, x_n]^T$. Suppose that $n \gtrsim \epsilon^{-4} d$. Then, the dataset X is ϵ-dispersed, i.e., $\mathscr{D}(X) \leq \epsilon$ and locally ϵ-dispersed, i.e., $\mathscr{D}(X - 1x_j^T) \leq \epsilon \ \forall j \in [n]$ with high probability.*

3.5.1 Dvoretzky's Theorem

In this section, we present a connection to Dvoretzky's theorem (Dvoretzky, 1959), a fundamental result in functional analysis and high-dimensional convex geometry (Vershynin, 2011).

Theorem 9 (Dvoretzky's Theorem)**.** *(Geometric version) Let $\mathcal{C}$ be a symmetric convex body in $\mathbb{R}^n$. For any $\epsilon > 0$, there exists an intersection $\mathcal{C}_S \triangleq \mathcal{C} \cap S$ of $\mathcal{C}$ by a subspace $S \subseteq \mathbb{R}^n$ of dimension $k(n, \epsilon) \to \infty$ as $n \to \infty$ such that*

$$(1-\epsilon)B_2 \subseteq \mathcal{C}_S \subseteq (1+\epsilon)B_2$$

where B_2 is the n-dimensional Euclidean unit ball.

The above shows that there exists a k-dimensional linear subspace such that the intersection of the convex body $\mathcal{C}$ with this subspace is approximately spherical; that is, it is contained in a ball of radius $1+\epsilon$ and contains a ball of radius $1-\epsilon$.

If we represent the linear subspace in Theorem 9 via the range of the matrix X and let $\mathcal{C}$ be the ℓ_1 ball, it is straightforward to show that Dvoretzky's theorem reduces to the isometry condition (6) up to a scalar normalization. Therefore, the isometry condition (6) can be interpreted as a condition to guarantee that the ℓ_1 ball is near-spherical when restricted to the range of the training data matrix.

3.6 Deep neural networks

Consider the deep neural network of L layers composed of sequential two-layer blocks considered in Section (2.1) as follows

$$f_\theta(x) = W^{(L)}\sigma(W^{(L-1)}\cdots W^{(3)}\sigma(W^{(2)}\sigma(W^{(1)}x)\cdots)), \quad \theta \triangleq (W^{(1)}, \cdots, W^{(L)}).$$

Theorems 12 and 13 in this section extend Theorem 4 to three-layer ReLU networks and derive their corresponding convex Lasso formulation over a larger discrete wedge product dictionary. We first illustrate that our results apply to neural networks of arbitrary depth.

3.6.1 ℓ_p regularization

Suppose that the number of layers, L, is even and consider the non-convex training problem

$$p^* \triangleq \min_{\theta} \ell(f_\theta(X), y) + \lambda \sum_{\ell=1}^{L/2} \sum_{j=1}^{m} \|W_{j\cdot}^{(2\ell-1)}\|_p^2 + \|W_{\cdot j}^{(2\ell)}\|_p^2 \,, \tag{22}$$

where $X \in \mathbb{R}^{n \times d}$ is the training data matrix, $y \in \mathbb{R}^n$ is a vector containing the training labels, l and $\lambda > 0$ is the regularization parameter. Here, $f(X)$ represents the output of the deep NN over the training data matrix X given by $f_\theta(X) = [f_\theta(x_1) \cdots f_\theta(x_n)]^T$. Note that the ℓ_p norms in the regularization terms are taken over the columns of odd layer weight matrices and rows of even weight matrices, which is consistent with the two-layer network objective in (2). When $p = 2$, the regularization term is the squared Frobenius norm of the weight matrices which reduces to the standard weight decay regularization term. We assume that the number of neurons in each layer is sufficiently large to meet the conditions required for applying Theorem 4 to two consecutive layers.

Theorem 10 (Structure of the optimal weights for ℓ_1 regularization)**.** *The weights of an optimal solution of (22) for $p = 1$ are given by*

$$\begin{aligned} W_j^{(1)} &= \alpha_j^{(1)} \star (x_{j_1^{(1)}} \wedge \cdots \wedge x_{j_{d-1}^{(1)}}) \,, \quad \text{and} \\ W_j^{(\ell)} &= \alpha_j^{(\ell)} \star (\tilde{x}_{j_1^{(\ell)}}^{(\ell)} \wedge \cdots \wedge \tilde{x}_{j_{d-1}^{(\ell)}}^{(\ell)}) \,, \quad \text{for } \ell = 3, 5, ..., L-1 \,, \end{aligned} \tag{23}$$

where $\alpha_j^{(\ell)}$ are scalar weights and $\tilde{x}_i^{(\ell)} \triangleq \sigma(W^{(\ell-1)} \cdots \sigma(W^{(1)} x_i) \cdots) \, \forall i$. Here, $W^{(\ell)}$ are optimal weights of the ℓ-th layer for the problem (22), and $j_k^{(\ell)} \in [n]$ are certain indices.

3.6.2 ℓ_2 regularization

Consider the training problem (22) with $p = 2$, which simplifies to

$$p^* \triangleq \min_{\theta} \ \ell(f_\theta(X), y) + \lambda \sum_{\ell=1}^{L} \|W^{(\ell)}\|_F^2 \,. \tag{24}$$

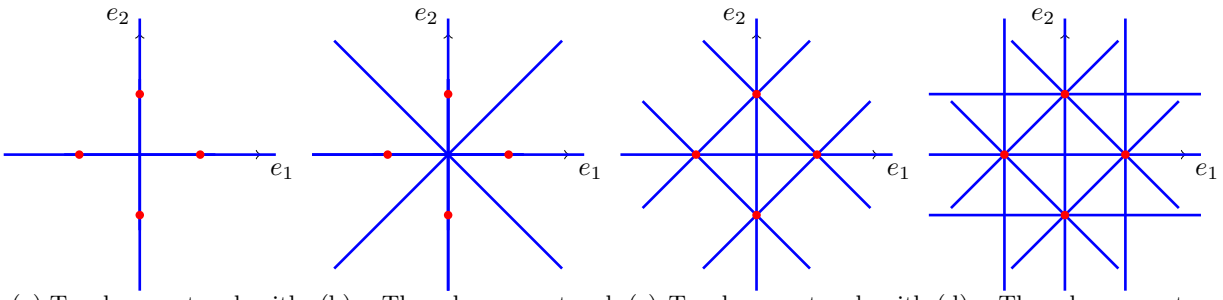
Theorem 11 (Structure of the optimal weights for ℓ_2 regularization)**.** *Consider an approximation of the optimal solution of (24) given by*

$$\begin{aligned} W_j^{(1)} &= \alpha_j^{(1)} \star (x_{j_1^{(1)}} \wedge \cdots \wedge x_{j_{d-1}^{(1)}}) \,, \quad \text{and} \\ W_j^{(\ell)} &= \alpha_j^{(\ell)} \star (\tilde{x}_{j_1^{(\ell)}}^{(\ell)} \wedge \cdots \wedge \tilde{x}_{j_{d-1}^{(\ell)}}^{(\ell)}) \,, \quad \text{for } \ell = 2, 3, ..., L \,, \end{aligned} \tag{25}$$

where $\alpha_j^{(\ell)}$ are scalar weights, $\tilde{x}_i^{(\ell)} \triangleq \sigma(W^{(\ell-1)} \cdots \sigma(W^{(1)} x_i) \cdots)$ and $j_k^{(\ell)} \in [n]$ are certain indices. The above weights provide the same loss as the optimal solution of (24). Moreover, the regularization term is only a factor $2/(1-\epsilon)$ larger than the optimal regularization term, where ϵ is an uppper-bound on the chamber diameters $\mathscr{D}(X^\ell)$ for $\ell = 0, ..., L-2$. Here, $X^\ell = \sigma(\cdots \sigma(X W^{(1)}) \cdots W^{(\ell-1)})$ are the ℓ-th layer activations of the network given by the weights (25).

3.6.3 Interpretation of the optimal weights

A fully transparent interpretation of how deep networks build representations can be given using our results. We have shown that each optimal neuron followed by a ReLU activation measures the positive distance of an



(a) Two-layer network without biases (b) Three-layer network without biases (c) Two-layer network with biases (d) Three-layer network with biases

Figure 7: Optimal space partitioning of two-layer and three-layer ReLU networks predicted by Theorems 4 and 13 for $p = 1$. The blue lines represent the breaklines of optimal neurons. The red dots represent the training data points. Theorem 13 is provided in Section 3.6.

input sample to the linear span (or affine hull, in the presence of bias terms) generated by a unique subset of training points using the formula

$$(x^T \star (x_{j_1} \wedge \ldots \wedge x_{j_k}))_+ = \mathbf{dist}_+\big(x, \mathbf{Span}(x_{j_1}, ..., x_{j_{d-1}})\big).$$

The ReLU activation serves as a crucial orientation determinant in this context. By nullifying negative signed distances, it effectively establishes a directionality in the space. Geometrically speaking, it delineates the specific side of the affine hull relevant for a particular input sample. In intermediate layers, the formula is applied to the activations of the previous layer, which are themselves signed distances to affine hulls of subsets of training data.

Since each layer of the network consists of a number of neurons, the activations of the network transforms the input data into a series of distances to these unique affine hulls as

$$\big[\mathbf{dist}_+\big(x, \mathbf{Span}(x_{j_1^{(1)}}, ..., x_{j_k^{(1)}})\big), \ldots, \mathbf{dist}_+\big(x, \mathbf{Span}(x_{j_1^{(m)}}, ..., x_{j_k^{(m)}})\big)\big].$$

Moreover, the information encapsulated within the weights of the network can be succinctly represented by the indices $j_1^{(1)}, \ldots, j_k^{(1)}, \ldots, j_1^{(m)}, \ldots, j_k^{(m)}$. These indices highlight the pivotal training samples that effectively determine the geometric orientation of each neuron. The formula essentially implies that the deep neural network's behavior and decisions are intrinsically tied to specific subsets of the training data, denoted by these critical indices.

This interpretation not only offers a geometric perspective on neural networks but also explains the pivotal role hidden layers play. Each hidden layer is a series of coordinate transformations, represented by the affine hulls of various data point subsets. As the data progresses through the network, it gets transformed and re-encoded, with every neuron contributing to this transformation based on its unique geometric connection to the training dataset.

3.6.4 Three-layer networks with bias-free first layer

We first consider the case when the first layer neurons $W_j^{(1)}$ are size $d \times 1$ $\forall j \in [m]$ for some arbitrary d, and the first layer neurons are bias-free, i.e., $b_j^{(1)} = 0 \, \forall j \in [m]$. We have the following theorem.

Theorem 12. *The three-layer neural network problem when $p = 1$, $W_j^{(1)} \in \mathbb{R}^{d \times 1}, b_j^{(1)} = 0 \, \forall j \in [m]$, set to zero is equivalent to the following convex Lasso problem*

$$\min_{z,b} \ell\big(K^{(1)} z_1 + K^{(2)} z_2 + 1_n b, y\big) + \lambda \|z\|_1. \tag{26}$$

The matrices $K^{(1)}$ *and* $K^{(2)}$ *are given by*

$$K_{ij}^{(1)} := \frac{\Big(\big(x_i \wedge \tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\big)_+ - \big(x_{j_0} \wedge \tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\big)_+\Big)_+}{\|\tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\|_1} \tag{27}$$

$$= \frac{\Big(\mathbf{Vol}_+\big(\mathcal{P}(x_i, \tilde{x}_{j_1}, ..., \tilde{x}_{j_{d-1}})\big) - \mathbf{Vol}_+\big(\mathcal{P}(x_{j_0}, \tilde{x}_{j_1}, ..., \tilde{x}_{j_{d-1}})\big)\Big)_+}{\|\tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\|_1} \tag{28}$$

and

$$K_{ij}^{(2)} := \frac{\Big(\mathbf{Vol}_+\big(\mathcal{P}(x_{j_0}, \tilde{x}_{j_1}, ..., \tilde{x}_{j_{d-1}})\big) - \mathbf{Vol}_+\big(\mathcal{P}(x_i, \tilde{x}_{j_1}, ..., \tilde{x}_{j_{d-1}})\big)\Big)_+}{\|\tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\|_1} \tag{29}$$

where $j = (j_0, j_1, ..., j_{d-1})$*. The multi-index* $(j_1, ..., j_{d-1})$ *indexes all combinations of* $d-1$ *vectors from the set* $\big\{\{x_i\}_{i=1}^n, \{x_i - x_j\}_{i=1,j=1}^n, \{e_k\}_{k=1}^d\big\}$ *and* $j_0 \in [n]$*. Each optimal first layer neuron weight* $w \in \mathbb{R}^d$ *satisfy equalities of the form*

$$x_i^T w = 0 \tag{30}$$

$$(x_i - x_j)^T w = 0 \tag{31}$$

$$e_k^T w = 0\,, \tag{32}$$

for a certain set of $i, j \in [n], k \in [d]$*. An optimal network can be constructed as follows:*

$$f(x) = \sum_j z_{1j}^* \frac{\Big(\big(x \wedge \tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\big)_+ - \big(x_{j_0} \wedge \tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\big)_+\Big)_+}{\|\tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\|_1} \tag{33}$$

$$+ \sum_j z_{2j}^* \frac{\Big(\big(x_{j_0} \wedge \tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\big)_+ - \big(x \wedge \tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\big)_+\Big)_+}{\|\tilde{x}_{j_1} \wedge \ldots \wedge \tilde{x}_{j_{d-1}}\|_1}\,, \tag{34}$$

where z^* *is an optimal solution to* (26)*.*

Remark. In addition to the optimal neurons for the two-layer case (Theorem 4), here we obtain additional neurons orthogonal to a subset of the data points and their pairwise differences. See Figure 7(b) for an illustration.

3.6.5 Three-layer networks with biased neurons

We now consider the case when the first layer neurons $W_j^{(1)}$ are size $d \times 1$ $\forall j \in [m]$ for some arbitrary dimension d, and the all the three layers contain trainable bias terms. We have the following theorem.

Theorem 13. *Consider the three-layer neural network problem when* $p = 1$ *and* $W_j^{(1)} \in \mathbb{R}^{d\times 1}$ $\forall j \in [m]$*. Each optimal first layer neuron weight-bias pair* $(w, b) \in \mathbb{R}^d \times \mathbb{R}$ *satisfy equalities of the form*

$$(x_i - x_\ell)^T w = 0 \tag{35}$$

$$(x_i - x_j)^T w = 0 \tag{36}$$

$$e_k^T w = 0 \tag{37}$$

$$x_\ell^T w + b = 0\,, \tag{38}$$

for a certain set of $i, j, \ell \in [n], k \in [d]$*.*

Remark. In addition to the optimal neurons for the two-layer case (Theorem 16), here we obtain additional neurons whose breaklines are translations of the affine hull of a subset of the data points to certain other data points. See Figure 7(d) for an illustration.

Remark. We note that the optimization problem and optimal networks take a similar form as in Theorem 12, except that the x_i are replaced by $x_i - x_\ell$ and the bias term $b = -x_\ell^T w$ is added.

3.7 Space partitioning of optimal deep networks

We now illustrate the optimal two-layer neurons predicted by Theorems 4 and compare them with optimal three-layer neurons (see Theorems 13-12 in the Section 3.6) as regularization tends to zero for $p = 1$ unless stated otherwise. Consider the two-dimensional training data $\{x_1 = (1, 0), x_2 = (0, 1), x_3 = (-1, 0), x_4 = (0, -1)\}$ shown in Figure 7.

In panel (a), we consider a two-layer ReLU network without biases. Two optimal neurons are $(w_1^T x)_+$, $(w_2^T x)_+$, given by Theorem 2. Their breaklines, $w_1^T x = 0$ and $w_2^T x = 0$, are plotted as blue lines, and pass through the origin and data points, since the optimal neurons are scalar multiples of the Hodge duals of 1-blades formed by data points.

In panel (b), we consider a three-layer ReLU network without biases, and we display all four optimal first layer neurons given by Theorem 12. In addition to the neurons with breaklines $w_1^T x = 0$ and $w_2^T x = 0$, we also have $w_3^T x = 0$ and $w_4^T x = 0$ which are translations of the affine hulls, $\mathbf{Aff}(x_1, x_2)$ and $\mathbf{Aff}(x_2, x_3)$, to the origin.

In panel (c), we consider a two-layer ReLU network with biases regularized with $p = 2$, and we display all six optimal neurons given by Theorem 3. Their breaklines pass between each pair of samples, since the optimal neurons are scalar multiples of the Hodge duals of 1-blades formed by the differences of data points.

In panel (d), we consider a three-layer ReLU network with biases, and we display all 12 optimal first layer neurons given by Theorem 13. In addition to the breaklines that pass between each pair of samples, we also have translations of all possible affine combinations of size two, e.g., $\mathbf{Aff}(x_1, x_2)$, $\mathbf{Aff}(x_1, x_3)$,..., to every data point.

3.8 Vector-output Neural Networks

Consider the vector-output neural network problem in (2) given by

$$p_v^* \triangleq \min_{W^{(1)}, W^{(2)}, b} \ell\Big(\sum_{j=1}^m \sigma(X W_j^{(1)} + 1 b_j) W_j^{(2)}, Y \Big) + \lambda \sum_{j=1}^m \|W_j^{(1)}\|_p^2 + \|W_j^{(2)}\|_p^2. \tag{39}$$

Here, the matrix $Y \in \mathbb{R}^{n \times c}$ contains the c-dimensional training labels, and $W^{(1)} \in \mathbb{R}^{d \times m}$, $W^{(2)} \in \mathbb{R}^{m \times c}$, and $b \in \mathbb{R}^m$ are trainable weights. We have the following extension of the convex progam (14) for vector-output neural networks.

$$\hat{p}_v \triangleq \min_{Z \in \mathbb{R}^{p \times c}} \ell\big(KZ, Y\big) + \lambda \sum_{j=1} \|Z_j\|_2, \tag{40}$$

where Z_j is the j-th column of the matrix Z.

Theorem 14. *Define the matrix K as follows*

$$K_{ij} = \frac{\big(x_i \wedge x_{j_1} \wedge \cdots \wedge x_{j_{d-1}}\big)_+}{\|x_{j_1} \wedge \ldots \wedge x_{j_{d-1}}\|_p},$$

where the multi-index $j = (j_1, ..., j_{d-1})$ is over all combinations of r rows and $r = \mathbf{rank}(X)$. It holds that

- *when $p = 1$, the convex problem (40) is equivalent to the non-convex problem (39), i.e., $p_v^* = \hat{p}_v$.*
- *when $p = 2$, the convex problem (40) is a $\frac{1}{1-\epsilon}$ approximation of the non-convex problem (39), i.e., $p_v^* \leq \hat{p}_v \leq \frac{1}{1-\epsilon} p_v^*$, where $\epsilon \in (0, 1)$ is an upper-bound on the maximum chamber diameter $\mathscr{D}(X)$.*

An neural network achieving the above approximation bound can be constructed as follows:

$$f(x) = \sum_j Z_j^* \frac{\big(x_i \wedge x_{j_1} \wedge \cdots \wedge x_{j_{d-1}}\big)_+}{\|x_{j_1} \wedge \ldots \wedge x_{j_{d-1}}\|_p}, \tag{41}$$

where Z^* is an optimal solution to (40).

4 Numerical Results

In this section, we introduce and examine a numerical procedure to take advantage of the closed-form formulas in refining neural network parameters and producing a geometrically interpretable network.

4.1 Refining neural network weights via geometric algebra

We apply the characterization of Theorem 4, which states that the hidden neurons are scalar multiples of $\star(x_{j_1} \wedge \cdots \wedge x_{j_{d-1}})$, Additionally, they are orthogonal to the $r-1$ training data points specified by $x_{j_1}, \cdots, x_{j_{d-1}}$, where r represents the rank of the training data matrix.

The inherent challenge lies in identifying the specific subset of the $r-1$ training points needed to form each neuron. Fortunately, this subset can be estimated when we have access to approximate neuron weights, typically acquired using standard non-convex heuristics such as stochastic gradient descent (SGD) or variants such as Adam and AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2018). After obtaining an approximate weight vector for each neuron, we can gauge which subsets of training data are nearly orthogonal to the neuron. This is achieved by evaluating the inner-products between the neuron weight and all training vectors, subsequently selecting the $r-1$ entries of the smallest magnitude. This refinement, which we term the *polishing* process, is delineated as follows for each neuron $w_1, ..., w_m$:

For each $j \in [m]$ (optional: Append 1 to the training samples to account for the neuron bias term)

1. Calculate the inner-product magnitudes: $|x_i^T w_j|$ for each $i \in [n]$.
2. Identify the $r-1$ training vectors with the minimal inner-product magnitude, denoted as $x_{j_1}, \cdots, x_{j_{d-1}}$.
3. Update the neuron using: $w_j \leftarrow \star(x_{j_1} \wedge \cdots \wedge x_{j_{d-1}}) = \times(x_{j_1}, \ldots, x_{j_{d-1}})$. As a result, we have $w_j \perp x_{j_1}, \ldots, x_{j_{d-1}}$. This can be done by solving the linear system $w_j^T x_{j_i} = 0$ for $i = 1, ..., r-1$, or finding a minimal left singular vector of the matrix $[x_{j_1}, \ldots, x_{j_{d-1}}]$, and normalizing w_j such that $\|w_j\|_p = 1$.
4. Optimize the weights of the following layer(s).
5. Optimize the scaling factors between consecutive layers (see Supplementary Material 9.13.1).

As a result, each neuron is assigned a closed-form symbolic expression, which only depends on a small subset of training samples.

4.1.1 Computational complexity of polishing

For a dataset of n samples of dimension d, the cost of applying the polishing process to a layer of m neurons is given by $O(mnd) + O(md^\omega)$, where ω is the matrix multiplication exponent, e.g., $\gamma = 3$ using classical solvers (Gloub and Van Loan, 1996) and $\omega \leq 2.376$ using fast matrix multiplication. We note that the latter class of algorithms are not practical for realistic sizes. However, $O(md^\omega)$ can be reduced to $O(\sqrt{\kappa} md^2 \log(1/\epsilon))$ for ϵ-approximate solutions of the linear system, where κ is the condition number. The computational cost is dominated by the calculation of inner products ($O(mnd)$) for large n and the $d \times d$ linear system solve ($O(md^\omega)$) for large d.

4.1.2 Toy spiral dataset

Figure 8 for the 2D spiral dataset and a two-layer neural network optimized with squared loss. In the initial panel of this figure, the training curve of a two-layer ReLU neural network from (2) is depicted, considering $p = 2$ and weight decay regularization set at $\beta = 10^{-5}$. The dataset, divided into two classes represented by blue and red crosses, is showcased in the second panel. By resorting to the dual formulation in (5), the global optimum value is computed. Notably, while SGD is far from the global optimum, the *polishing* process enhances the neurons, leading to a marked improvement in the objective value—evidenced by the solid line in the left panel. A comparative visualization of the decision region pre and post-polishing is presented in the subsequent panels, highlighting the enhanced data distribution fit due to the polishing.

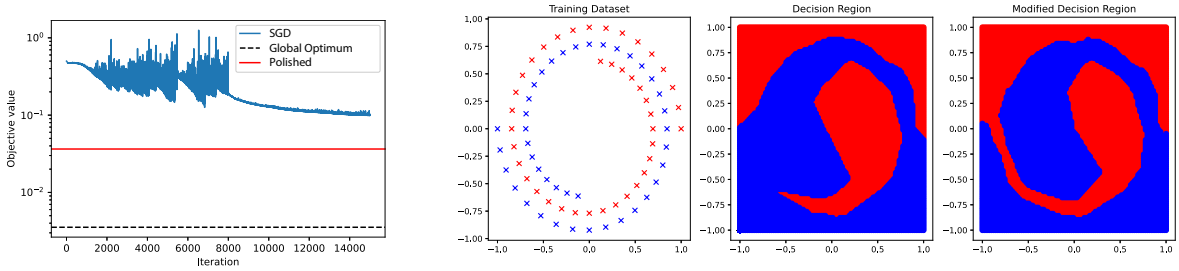
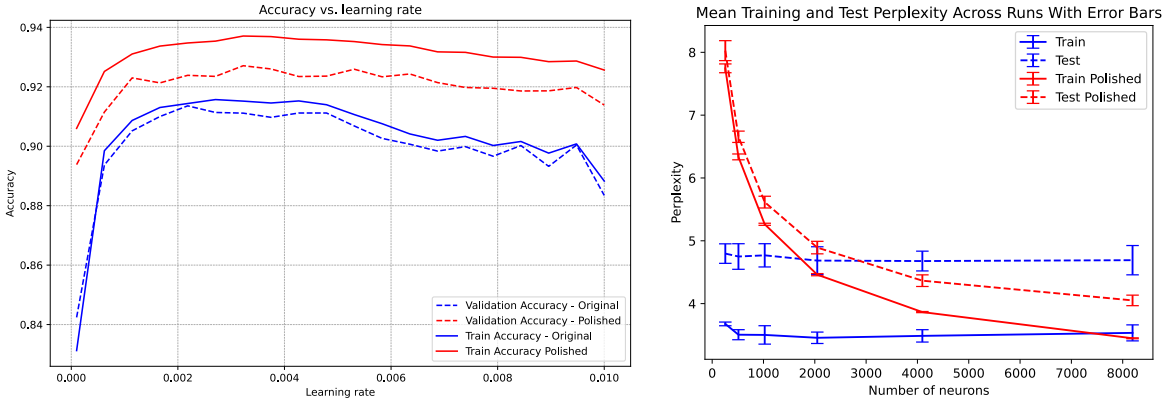


Figure 8: Comparison of SGD and polishing via geometric algebra in the spiral dataset.



(a) 4-layer convolutional neural network

(b) 2-layer autoregressive language model

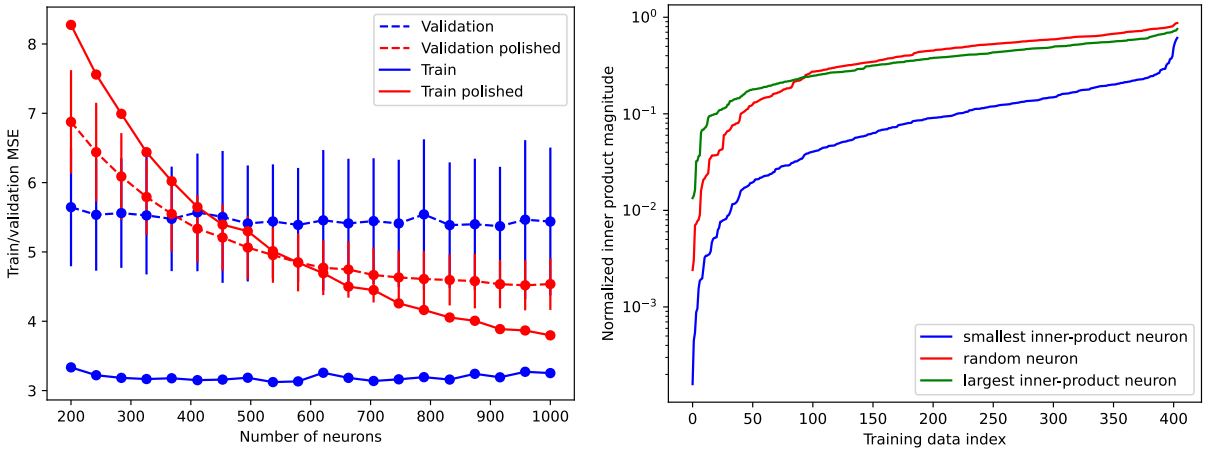
Figure 9: Comparison of AdamW and polishing CIFAR image classification (a) character-level MLP trained on a small subset of Wikipedia (b). Section 7.1 in the Supplementary Material contains additional numerical results, including a comprehensive hyperparameter search.

4.1.3 Real datasets

To illustrate the polishing strategy, we present three examples in Figures 9 and Figure 10.

In Figure 9 (a), we investigate binary image classification on the CIFAR dataset (Krizhevsky and Hinton, 2009). A four-layer convolutional network composed of two convolutional layers with $3 \times 3 \times 32$ filters and two fully connected layers with 512 hidden neurons is trained to distinguish class 0 (airplane) from class 2 (bird). We train it via AdamW optimizer using default hyperparameters and varying learning rate using 20 epochs and a batch size of 2048. After training, we implement the proposed polishing process on the first layer weights. Next, we re-train the second-layer weights while first layer weights are fixed via convex optimization. The resulting average train/validation accuracies are plotted over 5 independent trials to account for the randomness in optimization. We observe that the polishing process improves both the training and test accuracy. In Section 7.1.1, we provide additional results with different hyperparameters.

In Figure 9 (b), repeat the same polishing strategy for a small character-based autoregressive language model. We train a two-layer ReLU network to predict the next character in a sequence of characters from a small subset of Wikipedia consisting of first 650000 characters from the article titled 'Neural network (machine learning)' and other articles linked from the same page. We use the AdamW optimizer with a learning rate of 10^{-4} and a batch size of 8192. The block size is set to 16 characters. We apply polishing to the first layer weights. Next, we re-optimize the final layer weights while the first layer weights are fixed via convex optimization. The resulting average train/validation accuracies, along with 1-standard deviation error bars, are plotted over 8 independent trials to account for the randomness in optimization. We observe a significant improvement in perplexity after polishing when the number of neurons is large enough. In Section 7.1.5, we provide additional results with different hyperparameters.



(a) AdamW and polishing train/validation errors

(b) Distribution of inner-product magnitudes

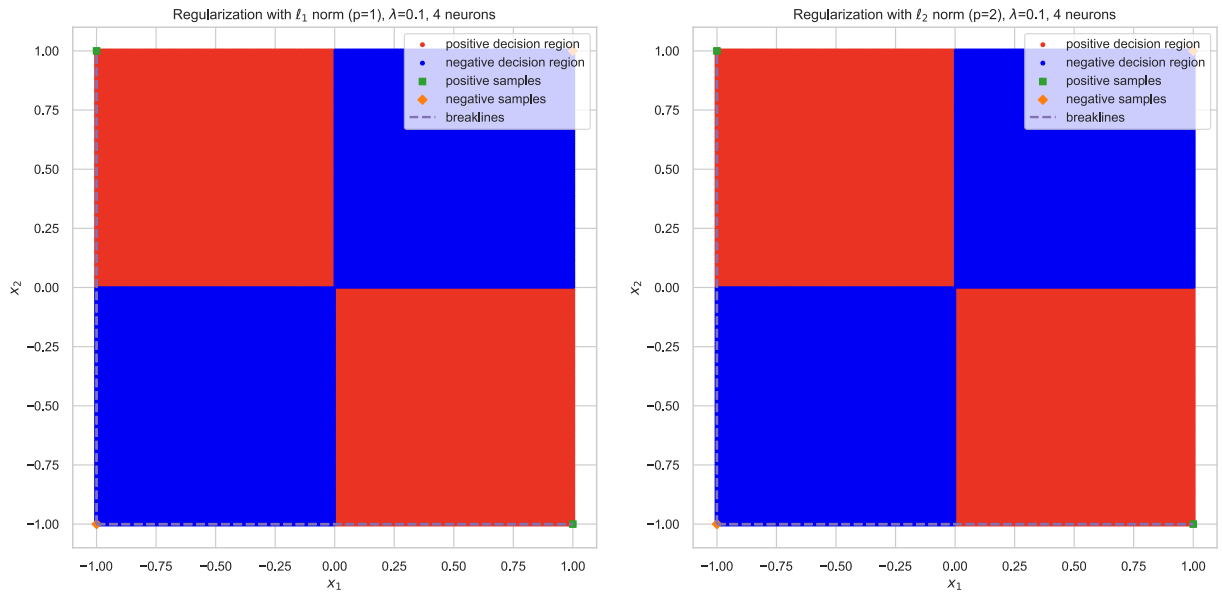
Figure 10: AdamW and polishing via geometric algebra in the Boston Housing dataset.

In Figure 10, we demonstrate the polishing strategy applied within a tabular learning context using the Boston Housing dataset. This dataset consists of 506 samples, each with 13 features representing various attributes of housing in Boston. We use a two-layer neural network with varying number of hidden neurons in the first layer, trained to predict the median value of owner-occupied homes. The network is trained using the AdamW optimizer with a learning rate of 10^{-2} and a batch size of 16 over 100 epochs. After training, we apply the proposed polishing process to the first layer, followed by re-optimizing the second-layer weights. The resulting train and validation MSE as well as one standard devation error bars are plotted over 100 independent trials to account for the randomness in optimization. As shown in Figure 10 (a), the polishing process consistently results in improved performance when the number of neurons are sufficiently large, similar to previous cases, demonstrating its robustness across different types of datasets. In Figure 10 (b), we present the distribution of the magnitude of inner-products between the weight vectors of the AdamW-trained first layer before polishing. This plot shows that many inner products are small, on the order of 10^{-2} to 10^{-3}, when the vectors are normalized to have a unit Euclidean norm. In Figure 39 in Section 7.6 of the Appendix, we present additional plots for traditional methods, showing that they perform worse compared to the polishing process. The work Adlam et al. (2020) reports the validation accuracy of kernel ridge regression, NN ensembles and Bayesian NNs, which all underperform compared to our approach. In the Supplementary Material (Section 7.1), we provide a detailed analysis of the effect of changing the hyperparameters and optimizers, including the learning rate, momentum parameters (β_1 and β_2 in Adam and AdamW), batch sizes, number of epochs, and also present additional results with fully connected networks and other binary classification tasks. We observe that the polishing process consistently improves the quality of the weights, leading to a significant improvement in the accuracy of the network while making the neurons fully interpretable as oriented distance functions via geometric algebra.

4.2 Comparison of ℓ_2 and ℓ_1 regularized neural networks

In this section we compare the predictions of optimal neural networks with p-regularized neurons, specifically focusing on $p = 2$ and $p = 1$. In Figure 11, we compare the optimal decision boundaries of ℓ_1 and ℓ_2 regularized NNs on the XOR dataset with $n = 4$ samples by solving the dual convex problem in (6). It can be seen that both models yield the same decision region. Moreover, two distinct breaklines of 4 optimal neurons are plotted. These can be identified as the affine hulls of a subset of training points.

In Figure 12, we compare ℓ_1 and ℓ_2 regularized neural networks on the spiral dataset across varying levels of regularization strength λ by solving the convex Lasso formulations. It can be observed that the optimal



(a) $p = 1$ (b) $p = 2$

Figure 11: Comparison of ℓ_1 and ℓ_2 regularized neural networks on the XOR dataset. The breaklines of the optimal neurons are shown as dashed lines, representing the affine hulls $\mathbf{Aff}\left(\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}\right)$ and $\mathbf{Aff}\left(\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}\right)$ which pass through a subset of the training data points.

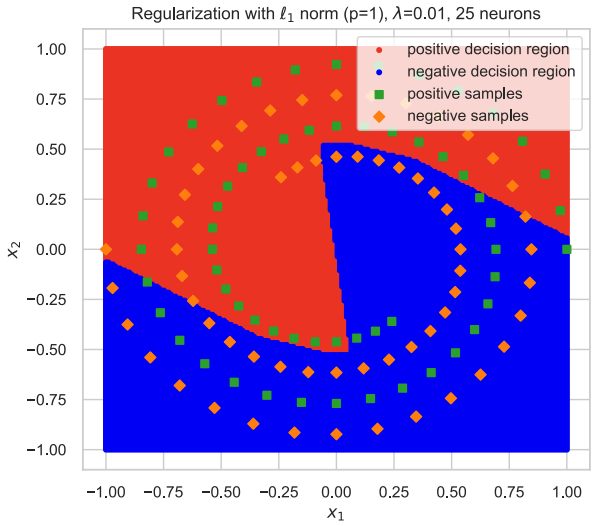
decision regions differ for large values of λ, whereas they become nearly identical for small values of λ. This is expected since with small λ, the network is forced to interpolate the data points regardless of the type of the regularization norm.

5 Discussion

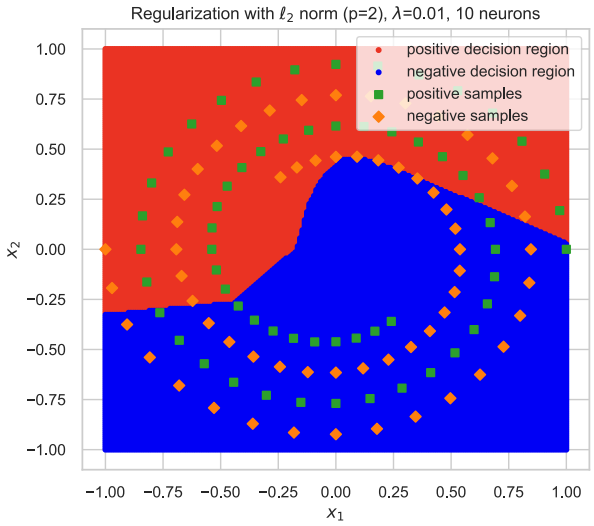
In this work, we have presented an analysis that uncovers a deep connection between Clifford's geometric algebra and optimal ReLU neural networks. By demonstrating that optimal weights of such networks are intrinsically tied to the wedge product of training samples, our results enable an understanding of how neural networks build representations as explicit functions of the training data points. Moreover, these closed-form functions not only provide a theoretical lens to understand neural networks, but also has the potential to guide new architectures and training algorithms that directly harness these geometric insights.

Computational complexity of global optimization

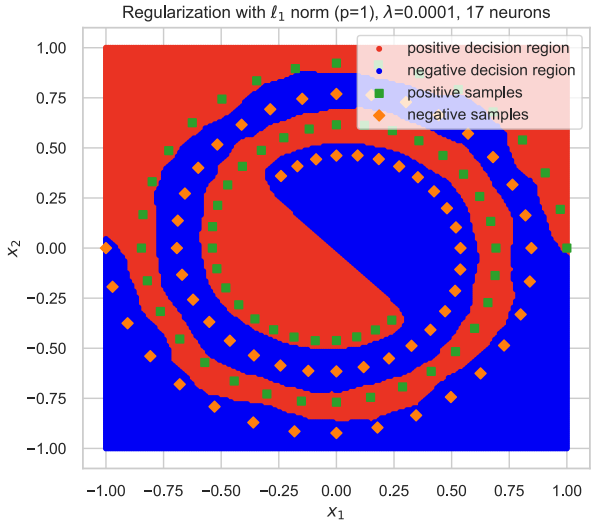
The computational complexity of the polishing process is dominated by step 3, which involves solving a linear system or finding a minimal left singular vector. This can be done in $O(n^2 r)$ time using the QR decomposition or the SVD. This process is repeated for each neuron, resulting in a total complexity of $O(n^2 rm)$, where m is the number of neurons. In contrast, the complexity of training the neural network to global optimal using the convex programs derived in Theorems 4 is $O(\binom{n}{r} n^2)$, which is tractable for small r. Note that for convolutional neural networks, the rank is bounded by the spatial size of the filter, which is a small constant (Ergen and Pilanci, 2024). Another application where the data is inherently low rank is Neural Radiance Fields (Mildenhall et al., 2021). The exponential complexity in r can not be improved unless $P = NP$ (Pilanci and Ergen, 2020; Wang and Pilanci, 2023). However, the convex programs can be well-approximated by sampling the wedge products, in a similar manner to the randomized sampling employed in convex formulations of NNs (Ergen and Pilanci, 2024; 2023; Mishkin et al., 2022). Recent work showed that random sampling of polynomially many variables in the convex program (5) provides a strong approximation with only logarithmic gap to the global optimum (Kim and Pilanci, 2024). Another work



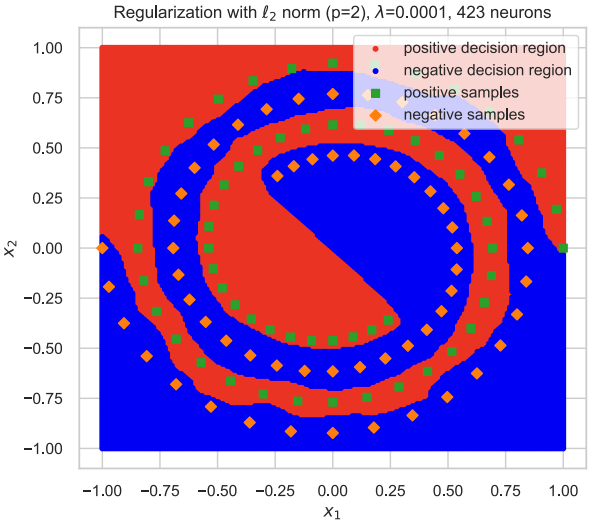
(a) $p = 1$ and $\lambda = 10^{-2}$



(a) $p = 2$ and $\lambda = 10^{-2}$



(a) $p = 1$ and $\lambda = 10^{-4}$



(a) $p = 2$ and $\lambda = 10^{-4}$

Figure 12: Comparison of ℓ_1 and ℓ_2 regularized neural networks on the spiral dataset. Note that the optimal decision boundaries are different for large values of λ but become very similar for small values of λ.

(Wang et al., 2024) introduced randomized algorithms for geometric algebra, which is a promising direction to make progress in this area.

Interpretability

Our findings also contribute to the broader challenge of neural network interpretability. The polishing process is expected to improve the quality of the weights, leading to a significant improvement in the accuracy of the network while making the neurons fully interpretable as oriented distance functions via geometric algebra. More precisely, after polishing each ReLU neuron precisely outputs $(x_i^T w)_+ = \mathbf{dist}_+\big(x_i, \mathbf{Span}(x_{j_1}, ..., x_{j_{d-1}})\big)$. This representation is similar to that of Support Vector Machines, where the model is defined by a weighted combination of training samples. By elucidating the roles hidden layers play in encoding geometric information of training data points through signed volumes, we have taken a step towards a more transparent and foundational theory of deep learning.

Uniqueness

We note that the optimal weights of a ReLU neural network are not unique, and permutation, merging and splitting operations on the neurons can lead to equivalent networks. However, all globally optimal solutions can be recovered via the set of optimal solutions of the convex program (5) by considering these three operations (Mishkin and Pilanci, 2023; Wang et al., 2021). Moreover, under certain assumptions, the convex program for univariate data admits a unique solution (Boursier and Flammarion, 2023). In addition, all stationary points of the non-convex training objective can be recovered via the convex program when certain variables are constrained to be zero (Wang et al., 2021), up to permutation, merging and splitting. An important open question is characterizig the entire optimal set of the convex programs via geometric algebra, which we leave as future work.

Other architectures

Our findings can be extended to several other widely used network architectures. For instance, convolutional neural networks can be transformed into fully connected networks by reshuffling the data matrix, as shown in Ergen and Pilanci (2021d). Some of our results can be directly applied by redefining the training data vectors. This can be extended to deep CNNs with short receptive fields (Brendel and Bethge, 2019). For other generic CNNs, the results can be extended by employing the same approach. Additionally, we believe that our results can be extended to transformer architectures employing linear or ReLU attention, as convex formulations of these networks have been analyzed in Sahiner et al. (2022). Further results for various other neural network architectures are provided in Section 8 of the Appendix.

There are many other open questions for further research. Exploring how these insights apply to state-of-the-art network architectures, or in the context of different regularization techniques and variations of activation functions, such as the ones in attention layers, could be of significant interest. While our techniques allow for the interpretation of layer weights in popular pretrained network models, we leave this for further research. Additionally, practical implications of our results, including potential improvements to the polishing process remain to be fully explored. Our results also underlines the potential and utility of integrating geometric algebra into the theory of deep learning.

References

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Adityanarayanan Radhakrishnan, Mikhail Belkin, and Caroline Uhler. Wide and deep neural networks achieve consistency for classification. *Proceedings of the National Academy of Sciences*, 120(14): e2208779120, 2023.

Yoshua Bengio, Nicolas Roux, Pascal Vincent, Olivier Delalleau, and Patrice Marcotte. Convex neural networks. *Advances in neural information processing systems*, 18, 2005.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems*, 32, 2019.

Stephen D Fisher and Joseph W Jerome. Spline solutions to l1 extremal problems in one and several variables. *Journal of Approximation Theory*, 13(1):73–83, 1975.

Enno Mammen and Sara Van De Geer. Locally adaptive regression splines. *The Annals of Statistics*, 25(1): 387–413, 1997.

Randall Balestriero et al. A spline theory of deep learning. In *International Conference on Machine Learning*, pages 374–383. PMLR, 2018.

Jason M Klusowski and Andrew R Barron. Approximation by combinations of relu and squared relu ridge functions with l1 and l0 controls. *IEEE Transactions on Information Theory*, 64(12):7649–7656, 2018.

Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? In *Conference on Learning Theory*, pages 2667–2690. PMLR, 2019.

Rahul Parhi and Robert D Nowak. Banach space representer theorems for neural networks and ridge splines. *Journal of Machine Learning Research*, 22(43):1–40, 2021.

Rahul Parhi and Robert D Nowak. What kinds of functions do deep neural networks learn? insights from variational spline theory. *SIAM Journal on Mathematics of Data Science*, 4(2):464–489, 2022.

Francesca Bartolucci, Ernesto De Vito, Lorenzo Rosasco, and Stefano Vigogna. Understanding neural networks with reproducing kernel banach spaces. *Applied and Computational Harmonic Analysis*, 62:194–236, 2023.

Michael Unser. A representer theorem for deep neural networks. *Journal of Machine Learning Research*, 20 (110):1–30, 2019.

Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pages 7695–7705. PMLR, 2020.

Tolga Ergen and Mert Pilanci. Convex geometry and duality of over-parameterized neural networks. *The Journal of Machine Learning Research*, 22(1):9646–9708, 2021a.

Burak Bartan and Mert Pilanci. Training quantized neural networks to global optimality via semidefinite programming. In *International Conference on Machine Learning*, pages 694–704. PMLR, 2021a.

Tolga Ergen and Mert Pilanci. Revealing the structure of deep neural networks via convex duality. In *International Conference on Machine Learning*, pages 3004–3014. PMLR, 2021b.

Tolga Ergen and Mert Pilanci. Global optimality beyond two layers: Training deep relu networks via convex programs. In *International Conference on Machine Learning*, pages 2993–3003. PMLR, 2021c.

Yifei Wang and Mert Pilanci. The convex geometry of backpropagation: Neural network gradient flows converge to extreme points of the dual convex program. In *International Conference on Learning Representations*, 2022.

Yifei Wang, Jonathan Lacotte, and Mert Pilanci. The hidden convex optimization landscape of regularized two-layer relu networks: an exact characterization of optimal solutions. In *International Conference on Learning Representations*, 2021.

Tolga Ergen, Arda Sahiner, Batu Ozturkler, John Pauly, Morteza Mardani, and Mert Pilanci. Demystifying batch normalization in relu networks: Equivalent convex optimization models and implicit regularization. *International Conference on Learning Representations*, 2022a.

Jonathan Lacotte and Mert Pilanci. All local minima are global for two-layer relu neural networks: The hidden convex optimization landscape. *arXiv e-prints*, 2020.

Burak Bartan and Mert Pilanci. Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time. *arXiv preprint arXiv:2101.02429*, 2021b.

Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. Activation atlas. *Distill*, 4 (3):e15, 2019.

David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

Laura O'Mahony, Vincent Andrearczyk, Henning Müller, and Mara Graziani. Disentangling neuron representations with concept vectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3769–3774, 2023.

Emil Artin. *Geometric algebra*. Courier Dover Publications, 2016.

Chris Doran and Anthony Lasenby. *Geometric algebra for physicists*. Cambridge University Press, 2003.

Leo Dorst, Chris Doran, and Joan Lasenby. *Applications of geometric algebra in computer science and engineering*. Springer Science & Business Media, 2012.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Tolga Ergen, Halil Ibrahim Gulluk, Jonathan Lacotte, and Mert Pilanci. Globally optimal training of neural networks with threshold activation functions. In *The Eleventh International Conference on Learning Representations*, 2022b.

Rahul Parhi and Robert D Nowak. The role of neural network activation functions. *IEEE Signal Processing Letters*, 27:1779–1783, 2020.

Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

Emi Zeger, Yifei Wang, Aaron Mishkin, Tolga Ergen, Emmanuel Candès, and Mert Pilanci. A library of mirrors: Deep neural nets in low dimensions are convex lasso models with reflection features. *arXiv preprint arXiv:2403.01046*, 2024.

Etienne Boursier and Nicolas Flammarion. Penalising the biases in norm regularisation enforces sparsity. *Advances in Neural Information Processing Systems*, 36:57795–57824, 2023.

Yaniv Plan and Roman Vershynin. Dimension reduction by random hyperplane tessellations. *Discrete & Computational Geometry*, 51(2):438–461, 2014.

Yehoram Gordon. Some inequalities for gaussian processes and applications. *Israel Journal of Mathematics*, 50:265–289, 1985.

Aryeh Dvoretzky. A theorem on convex bodies and applications to banach spaces. *Proceedings of the National Academy of Sciences*, 45(2):223–226, 1959.

Roman Vershynin. Lectures in geometric functional analysis. *Unpublished manuscript. Available at http://www-personal. umich. edu/romanv/papers/GFA-book/GFA-book. pdf*, 3(3):3–3, 2011.

Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Gene H Gloub and Charles F Van Loan. Matrix computations. *Johns Hopkins Universtiy Press, 3rd edtion*, 1996.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, ON, Canada, 2009.

Ben Adlam, Jaehoon Lee, Lechao Xiao, Jeffrey Pennington, and Jasper Snoek. Exploring the uncertainty properties of neural networks' implicit priors in the infinite-width limit. *arXiv preprint arXiv:2010.07355*, 2020.

Tolga Ergen and Mert Pilanci. Path regularization: A convexity and sparsity inducing regularization for parallel relu networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Yifei Wang and Mert Pilanci. Polynomial-time solutions for relu network training: A complexity classification via max-cut and zonotopes. *arXiv preprint arXiv:2311.10972*, 2023.

Tolga Ergen and Mert Pilanci. The convex landscape of neural networks: Characterizing global optima and stationary points via lasso models. *arXiv preprint arXiv:2312.12657*, 2023.

Aaron Mishkin, Arda Sahiner, and Mert Pilanci. Fast convex optimization for two-layer relu networks: Equivalent model classes and cone decompositions. In *International Conference on Machine Learning*, pages 15770–15816. PMLR, 2022.

Sungyoon Kim and Mert Pilanci. Convex relaxations of relu neural networks approximate global optima in polynomial time. *arXiv preprint arXiv:2402.03625*, 2024.

Yifei Wang, Sungyoon Kim, Paul Chu, Indu Subramaniam, and Mert Pilanci. Randomized geometric algebra methods for convex neural networks. *arXiv preprint arXiv:2406.02806*, 2024.

Aaron Mishkin and Mert Pilanci. Optimal sets and solution paths of relu networks. In *International Conference on Machine Learning*, pages 24888–24924. PMLR, 2023.

Tolga Ergen and Mert Pilanci. Implicit convex regularizers of cnn architectures: Convex optimization of 2- and 3-layer networks in polytime. *International Conference on Learning Representations*, 2021d.

Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

Arda Sahiner, Tolga Ergen, Batu Ozturkler, John Pauly, Morteza Mardani, and Mert Pilanci. Unraveling attention via convex duality: Analysis and interpretations of vision transformers. In *International Conference on Machine Learning*, pages 19050–19088. PMLR, 2022.

Marcel Berger. *Geometry I*. Springer Science & Business Media, 2009.

Eugene Gover and Nishan Krikorian. Determinants and the volumes of parallelotopes and zonotopes. *Linear Algebra and its Applications*, 433(1):28–40, 2010.

Arda Sahiner, Tolga Ergen, John M Pauly, and Mert Pilanci. Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms. In *International Conference on Learning Representations*, 2020.

Werner Fenchel and Donald W Blackett. *Convex cones, sets, and functions*. Princeton University, Department of Mathematics, Logistics Research Project, 1953.

Hermann Weyl. The elementary theory of convex polyhedra. *Contributions to the Theory of Games*, page 3, 1950.

Yifei Wang, Tolga Ergen, and Mert Pilanci. Parallel deep neural networks have zero duality gap. In *The Eleventh International Conference on Learning Representations*, 2022.