Randomized Embeddings and Neural Networks

Mert Pilanci

October 9, 2023

Electrical Engineering

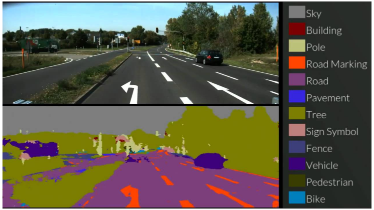
Stanford University

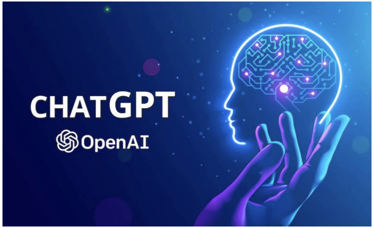
Sketching and Algorithm Design Workshop

Simons Institute for the Theory of Computing, UC Berkeley

Outline

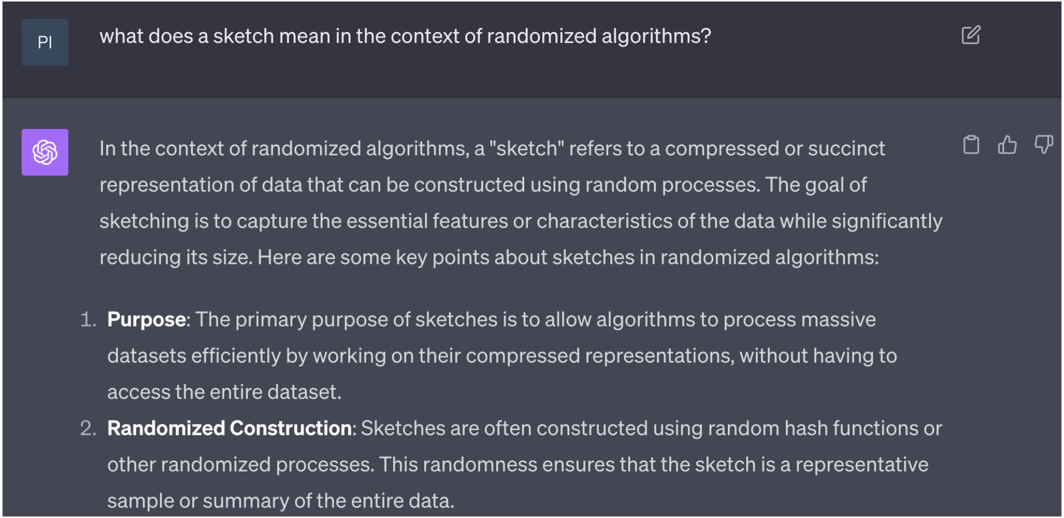
The impact of deep neural networks





[Y. LeCun, Y. Bengio, G. Hinton 2015]

ChatGPT



PI

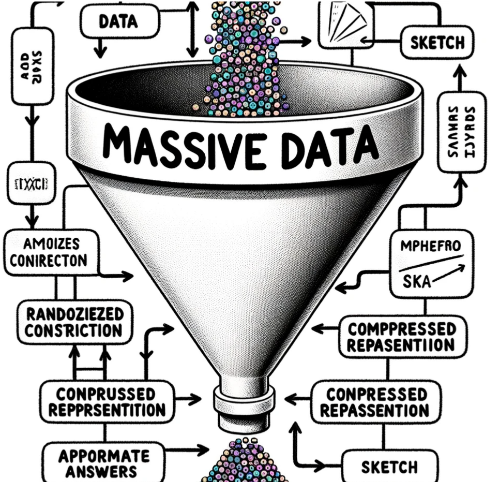
what does a sketch mean in the context of randomized algorithms?

In the context of randomized algorithms, a "sketch" refers to a compressed or succinct representation of data that can be constructed using random processes. The goal of sketching is to capture the essential features or characteristics of the data while significantly reducing its size. Here are some key points about sketches in randomized algorithms:

1. **Purpose**: The primary purpose of sketches is to allow algorithms to process massive datasets efficiently by working on their compressed representations, without having to access the entire dataset.
2. **Randomized Construction**: Sketches are often constructed using random hash functions or other randomized processes. This randomness ensures that the sketch is a representative sample or summary of the entire data.

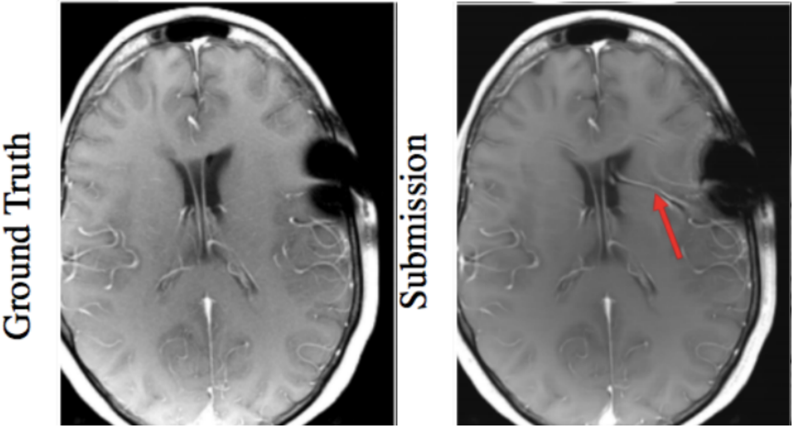
prompt: illustrate this using a diagram

reply (DALL-E 3): This illustration uses a funnel metaphor, where "Massive Data" enters the funnel and emerges as a "Sketch" at the bottom. Surrounding the funnel are words describing various properties of the sketching process.



Fast MRI Challenge, 2020

model generates a false vessel (Muckley et al.)



deep learning models

- are complex black-box systems based on non-convex optimization

  $\rightarrow$ **hard to interpret what the model is actually learning**
- often provide the best performance due to their large capacity

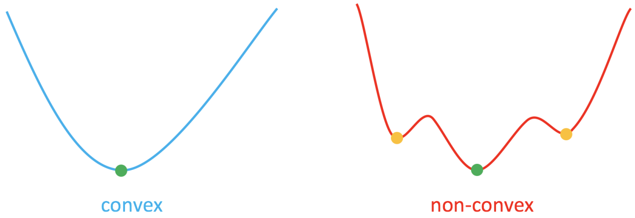
  $\rightarrow$ **challenging to train**

  GPT-3 has 175 billion parameters

  GPT-4 ?

  massive training data

Fundamental challenges in neural networks



- least-squares, logistic regression, support vector machines etc. are convex models, which can be solved efficiently and are understood well
- training a two-layer network to optimality is not achievable in polynomial-time (unless P=NP)

Open questions

- how to make training **energy/memory/data** efficient
- how to develop a **foundational theory** for neural networks
- how to **interpret** these models

Open questions

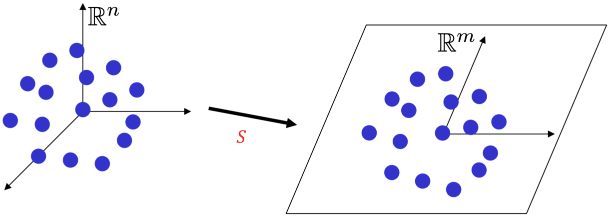
- how to make training **energy/memory/data** efficient
- how to develop a **foundational theory** for neural networks
- how to **interpret** these models

randomized embeddings play an important role in **all three** aspects

- **non-convex** neural networks problems can be converted to high-dimensional **convex** optimization
- **randomized embeddings** and **sketching** to reduce the dimension
- global optimality and approximation guarantees
- connections to **zonotopes** and **Clifford algebra**

Outline

1. Neural networks and current challenges
2. Randomized embeddings
3. Hidden convexity and randomization in neural networks
4. Clifford algebra



- let $V = \{z_1, ..., z_k\}$ be a set of points in $\mathbb{R}^n$ and let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a map where $m < n$
- we call f an **embedding** if $\|f(z_i) - f(z_j)\| \approx \|z_i - z_j\|$
- **Johnson-Lindenstrauss embeddings:** for $m \gtrsim \epsilon^{-2}\log(k)$ there exists a linear embedding $f(z) = Sz$, where $S \in \mathbb{R}^{m\times n}$ that approximately preserves distances:

$$(1-\epsilon)\|z_i - z_j\|_2^2 \le \|f(z_i) - f(z_j)\|_2^2 \le (1+\epsilon)\|z_i - z_j\|_2^2 \; \forall i, j \in [k] \tag{1}$$

Randomized Embeddings: $\ell_p \to \ell_q$

- generalization to arbitrary norms and arbitrary sets
- for $p, q \in [1, \infty)$ and $\epsilon \in (0, 1)$

$$(1-\epsilon)\|z\|_p \leq \|Sz\|_q \leq (1+\epsilon)\|z\|_p \quad \forall z \in \mathcal{V}$$

- $\mathcal{V}$ is any subset of $\mathbb{R}^n$

Randomized Embeddings: Subspace Embedding

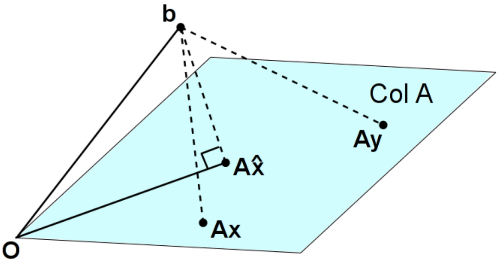
- let $p = q = 2$ and $\mathcal{V} = \mathbf{range}(A)$ for some fixed matrix $A \in \mathbb{R}^{n \times d}$
- randomly generate $S \in \mathbb{R}^{m \times n}$, e.g., i.i.d. ± 1, Gaussian,...

$$(1 - \epsilon)\|z\|_2 \leq \|Sz\|_2 \leq (1 + \epsilon)\|z\|_2 \quad \forall z \in \mathcal{V}$$

  holds with high probability when $m \gtrsim \epsilon^{-2}\textbf{rank}(A)$
- lengths of all vectors in the range of A are approximately preserved

Example: Least squares prediction

$$\min_x \|Ax - y\|_2^2$$



- A : n rows, d columns, $n \geq d$
- computational complexity

  Cholesky/QR: $O(nd^2)$

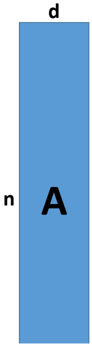
  Conjugate Gradient: $O\big(\sqrt{\kappa} nd \log(1/\epsilon)\big)$ for an ϵ-approximate solution

Example : Least squares prediction

- $A : n \times d$ feature matrix, and $y : n \times 1$ response vector
- original problem $\qquad \mathbf{OPT} = \min_x \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$

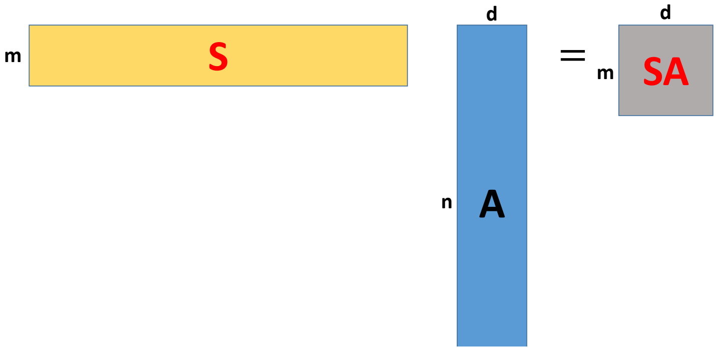
Example : Least squares prediction

- $A : n \times d$ feature matrix, and $y : n \times 1$ response vector
- original problem $\quad \mathbf{OPT} = \min_x \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$



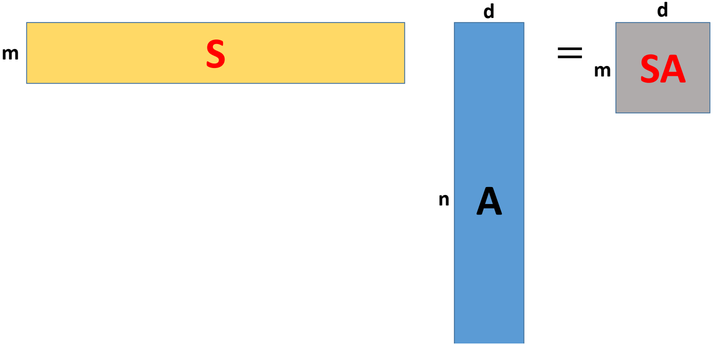
Example : Least squares prediction

- $A : n \times d$ feature matrix, and $y : n \times 1$ response vector
- original problem $\quad \mathbf{OPT} = \min_x \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$



Example : Least squares prediction

- $A : n \times d$ feature matrix, and $y : n \times 1$ response vector
- original problem $\quad \mathbf{OPT} = \min_x \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$
- approximation $\quad \widehat{x} = \arg\min_x \|S(Ax - y)\|_2^2$
- $S : m \times n$ **randomized embedding** (*sketching*) matrix (e.g., i.i.d. Gaussian)



Example : Least squares prediction

- $A : n \times d$ feature matrix, and $y : n \times 1$ response vector
- original problem $\quad \mathbf{OPT} = \min_x \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$
- approximation $\quad \widehat{x} = \arg\min_x \|S(Ax - y)\|_2^2$
- $S : m \times n$ **randomized embedding** (*sketching*) matrix (e.g., i.i.d. Gaussian)

m S

n A

= m SA

Õ(nd) $\quad$ **S = FD**

$\underbrace{F}_{\text{Restricted Isometry}}\underbrace{D}_{\text{Random diagonal } \pm 1}$

[Candès & Tao, 06; Krahmer & Ward 11]

Õ(nnz(A)) $\quad$ **sparse**

[Nelson & Nguyen 13; Bourgain, Dirksen, Nelson, 15]

Example : Least squares prediction

- $A : n \times d$ feature matrix, and $y : n \times 1$ response vector
- original problem $\mathbf{OPT} = \min_x \underbrace{\|Ax - y\|_2^2}_{f(Ax)}$
- approximation $\widehat{x} = \arg\min_x \|S(Ax - y)\|_2^2$
- $S : m \times n$ **randomized embedding** (*sketching*) matrix (e.g., i.i.d. Gaussian)

Theorem : Cost approximation

If $m \geq 1 + \mathrm{rank}(A)(1 + 1/\epsilon)$, then

$\mathbf{OPT} \leq f(A\widehat{x}) \leq (1+\epsilon)\mathbf{OPT}$

with high probability

[Sarlós 06; Rokhlin and Tygert 08; Pilanci and Wainwright, IEEE Trans. Info. Theory 2015; Bartan and Pilanci, IEEE Trans. Info. Theory 2023]

Example : Least squares prediction

Practical use

Airline dataset $n = 120,000,000, \ d = 28$

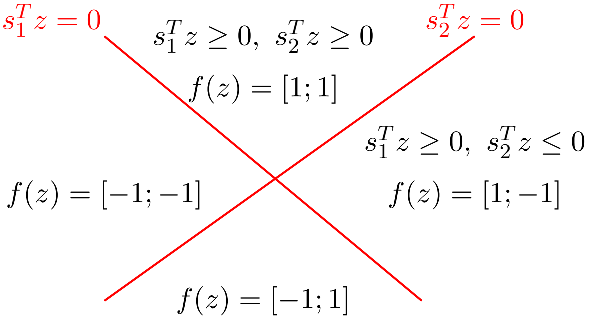
$m = 500$ gives 1.06-approximation

$m = 5000$ gives 1.006-approximation

[Pilanci and Wainwright, IEEE Trans. Info. Theory 2015; Bartan and Pilanci, IEEE Trans. Info. Theory 2023]

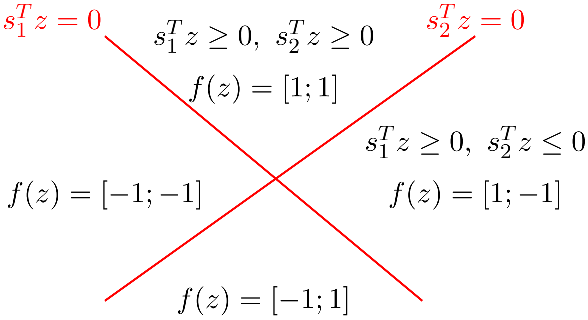
Quantized Embeddings: embeddings into to Hamming cube

- let $V = \{z_1, ..., z_k\} \subseteq \mathbb{S}^{n-1}$ be a set of points let $S \in \mathbb{R}^{m \times n}$ be i.i.d. Gaussian
- consider the map $f(z) := \mathbf{sign}(Sz)$



Quantized Embeddings: embeddings into to Hamming cube

- let $V = \{z_1, ..., z_k\} \subseteq \mathbb{S}^{n-1}$ be a set of points let $S \in \mathbb{R}^{m \times n}$ be i.i.d. Gaussian
- consider the map $f(z) := \mathbf{sign}(Sz)$



- generates **hyperplane arrangements** of the rows of S
- $\left| \frac{1}{m} \sum_{i=1}^m \mathbf{1}_{f(z) \neq f(z')} - \frac{1}{\pi} \cos^{-1}(z^T z') \right| \leq \epsilon \ \forall z, z' \in V$ w.h.p. if $m \gtrsim \epsilon^{-2} \log k$
- locality-sensitive hashing, one-bit compressed sensing and compressing large DNN models

[Charikar 2002; Boufounos & Baraniuk 08; Plan & Vershynin, 14, Saha & Srivastava & Pilanci, NeurIPS 2023]

Other problems where randomized embeddings are useful

- streaming setting: $A_{t+1} = A_t + \Delta_t \quad SA_{t+1} = SA_t + S\Delta_t$
- sketching features: $A \to AS$ is applying the left-sketch to the dual problem $(S^T A^T)$
- low-rank approximations of matrices and tensors
- sketch based preconditioners

Other problems where randomized embeddings are useful

- streaming setting: $A_{t+1} = A_t + \Delta_t \quad SA_{t+1} = SA_t + S\Delta_t$
- sketching features: $A \to AS$ is applying the left-sketch to the dual problem $(S^T A^T)$
- low-rank approximations of matrices and tensors
- sketch based preconditioners
- generic convex optimization problems $\min_{x \in C} f(Ax)$, including logistic regression, support vector machines, linear programs, semi-definite programs...

  Newton Sketch $((\nabla^2 f(x))^{1/2} S^T S (\nabla^2 f(x))^{1/2})^{-1} \nabla f(x)$ approximates Newton steps
- constrained/regularized problems: embedding dimension can be proportional to the **width** of the constraint set $\mathcal{C}$, i.e., $m \gtrsim \epsilon^{-2} \mathcal{W}(C)$

[Drineas & Kannan & Mahoney 2006; Rokhlin & Tygert 2008; Halko & Martinsson, & Tropp 2011; Wainwright & Pilanci 2016, 2017]

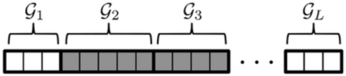
Least Squares with L1 regularization

$$\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$$

- L1 norm $\|x\|_1 = \sum_{i=1}^{d} |x_i|$

  encourages solution x^* to be sparse

Least squares with group L1 regularization



$$\min_x \left\| \sum_{i=1}^{L} A_i x_i - y \right\|_2^2 + \lambda \sum_{i=1}^{L} \|x_i\|_2$$

$\|x_i\|_2 = \sqrt{\sum_{j=1}^{d} x_{ij}^2}$

encourages solution x^* to be group sparse

most x_i^* blocks are zero

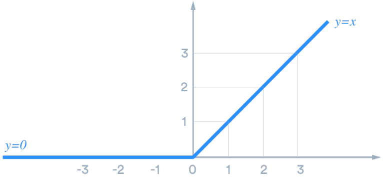
Outline

1. Neural networks and current challenges
2. Randomized embeddings
3. Hidden convexity and randomization in neural networks
4. Clifford algebra

Training two-layer neural networks: Non-convex optimization

$$p_{\text{non-convex}} := \underset{\substack{W_1 \in \mathbb{R}^{d\times m} \\ W_2 \in \mathbb{R}^{m\times 1}}}{\text{minimize}} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

where $\phi(u) = \max(0, u)$ is the ReLU activation



ReLU neural networks are equivalent to convex models

$$
\begin{aligned}
p_{\text{non-convex}} := \quad & \text{minimize} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right) \\
& W_1 \in \mathbb{R}^{d\times m} \\
& W_2 \in \mathbb{R}^{m\times 1} \\
p_{\text{convex}} := \quad & \text{minimize} \quad L\left(Z, y\right) + \lambda \underbrace{R(Z)}_{\text{convex regularization}} \\
& Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}
\end{aligned}
$$

[Pilanci & Ergen, ICML 2020; Neurips 2023]

$$
\begin{aligned}
p_{\text{non-convex}} := \quad & \underset{W_1 \in \mathbb{R}^{d\times m},\ W_2 \in \mathbb{R}^{m\times 1}}{\text{minimize}} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)
\end{aligned}
$$

$$
p_{\text{convex}} := \underset{Z \in \mathcal{K} \subseteq \mathbb{R}^{d\times p}}{\text{minimize}} \quad L\left(Z, y\right) + \lambda R(Z)
$$

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be obtained from an optimal solution to p_{convex}.

[Pilanci & Ergen, ICML 2020; Neurips 2023]

ReLU Network using squared loss = group Lasso using fixed features

data matrix $X \in \mathbb{R}^{n \times d}$ and label vector $y \in \mathbb{R}^n$ $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$

$$p_{\text{non-convex}} = \text{ minimize}_{W_1, W_2} \left\| \sum_{j=1}^{m} \phi(XW_{1j})W_{2j} - y \right\|_2^2 + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right)$$

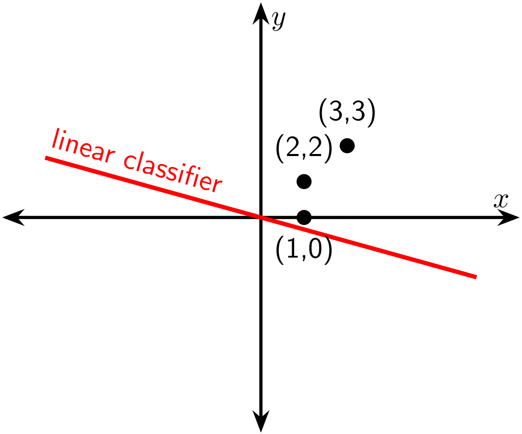
$$p_{\text{convex}} = \text{ minimize}_{u_1, v_1 \ldots u_p, v_p \in \mathcal{K}} \left\| \sum_{i=1}^{p} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|u_i\|_2 + \|v_i\|_2 \right)$$

$D_1, ..., D_p$ are fixed diagonal matrices

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be recovered from optimal non-zero u_i^*, v_i^*, $i = 1, ..., p$ as

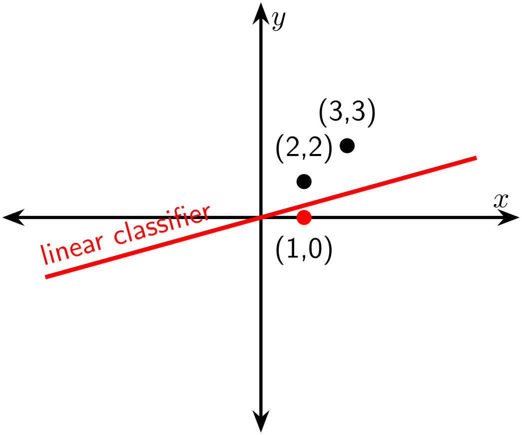
$W_{1j}^* = \frac{u_i^*}{\sqrt{\|u_i^*\|_2}}$, $W_{2j} = \sqrt{\|u_i^*\|_2}$ or $W_{1j}^* = \frac{v_j^*}{\sqrt{\|v_j^*\|_2}}$, $W_{2j} = -\sqrt{\|v_j^*\|_2}$.

$$n = 3 \text{ samples in } \mathbb{R}^d,\ d = 2 \quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

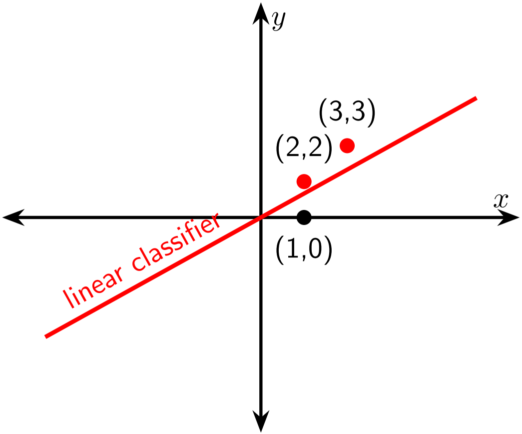
$$n = 3 \text{ samples in } \mathbb{R}^d,\ d = 2 \quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

$$D_2 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 0 & 0 \end{bmatrix}$$

$$n = 3 \text{ samples in } \mathbb{R}^d,\ d = 2 \quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$



$$D_1 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 1 & 0 \end{bmatrix}$$

$$D_2 X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} X = \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 0 & 0 \end{bmatrix}$$

$$D_4 X = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

Example: Convex Program for $n = 3, d = 2$

$$n = 3 \text{ samples} \quad X = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\min \left\| \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \end{bmatrix} (u_1 - v_1) + \begin{bmatrix} x_1^T \\ x_2^T \\ 0 \end{bmatrix} (u_2 - v_2) + \begin{bmatrix} 0 \\ 0 \\ x_3^T \end{bmatrix} (u_3 - v_3) - y \right\|_2^2$$

subject to

$$+ \lambda \Big(\sum_{i=1}^{3} \|u_i\|_2 + \|v_i\|_2 \Big)$$

$$D_1 X u_1 \geq 0, D_1 X v_1 \geq 0$$

$$D_2 X u_2 \geq 0, D_2 X v_2 \geq 0$$

$$D_4 X u_3 \geq 0, D_4 X v_3 \geq 0$$

equivalent to the non-convex two-layer NN problem

Computational Complexity

Learning two-layer ReLU neural networks with m neurons

$f(x) = \sum_{j=1}^{m} W_{2j}\phi(W_{j1}x)$

Previous result: ◦ Combinatorial $O(2^m n^{dm})$ (Arora et al., ICLR 2018)

$\boxed{\text{Convex program } O((\frac{n}{r})^r) \text{ where } r = \mathrm{rank}(X)}$

Computational Complexity

Learning two-layer ReLU neural networks with m neurons

$f(x) = \sum_{j=1}^{m} W_{2j}\phi(W_{j1}x)$

Previous result: ∘ Combinatorial $O(2^m n^{dm})$ (Arora et al., ICLR 2018)

Convex program $O((\frac{n}{r})^r)$ where $r = \mathrm{rank}(X)$

n : number of samples, d : dimension

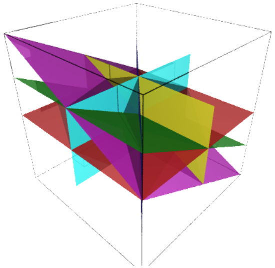
(i) polynomial in n and m for fixed rank r

(ii) exponential in d for full rank data $r = d$. Can not be improved unless $P = NP$ even for $m = 1$.

Number of variables = number of hyperplane arrangements

- convex program has at most $\left(\left(\frac{n}{r}\right)^r\right)$ variables

  #activation patterns of only **one neuron**
  $= \left|\{\mathbf{sign}(Xw)\ : w \in \mathbb{R}^d\}\right| \leq O\left(\left(\frac{n}{r}\right)^r\right)$ where $r = \mathbf{rank}(X)$.



- rank is constant for convolutional networks

  e.g., $3 \times 3 \times 1024$ convolution $\Longrightarrow$ $r = 9$ polynomial-time wrt all dims

How to approximately solve a high-dimensional convex problem

$$p_{\text{convex}} = \ \text{minimize}_{u_1,v_1\ldots u_p,v_p\in\mathcal{K}} \ \Big\| \sum_{i=1}^{p} D_i X(u_i - v_i) - y \Big\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|u_i\|_2 + \|v_i\|_2 \right)$$

- **idea:** randomly subsample variables $\{D_i X(u_i - v_i)\}_{i=1}^{p}$ to optimize, set the rest to zero

  goal: sample proportionally to some convenient **importance** measure

Sketching the convex neural network: subsampling variables

randomized algorithm

- sample $D_i = \mathrm{Diag}(Xu_i \geq 0)$ where u_i is i.i.d. Gaussian for $i = 1, ..., \tilde{p}$, $\tilde{p} \leq p$

  (**quantized random embedding / locality-sensitive hashing**)

- solve

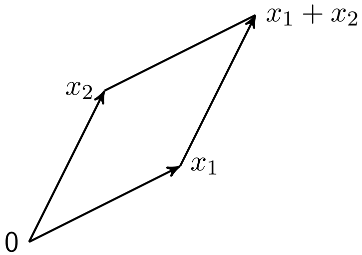
$$\text{minimize}_{u_1,v_1 \ldots u_{\tilde{p}},v_{\tilde{p}} \in \mathcal{K}} \left\| \sum_{i=1}^{\tilde{p}} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{\tilde{p}} \|u_i\|_2 + \|v_i\|_2 \right)$$

- construct neural network from the solution

Zonotopes

$$\mathcal{Z}(X) := \mathbf{conv}\Big\{ \sum_i x_i u_i, \quad u_i \in \{0,1\}\, \forall i \in [n] \Big\} = X^T[0,1]^n$$

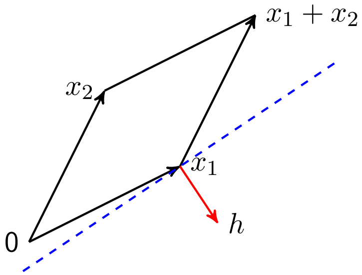
- $\mathcal{Z}\left(\begin{bmatrix} x_1^\intercal \\ x_2^\intercal \end{bmatrix}\right) = \mathbf{conv}\{0, x_1, x_2, x_1 + x_2\} \subseteq \mathbb{R}^2$



Zonotopes

$$\mathcal{Z}(X) := \mathbf{conv}\Big\{ \textstyle\sum_i x_i u_i, \quad u_i \in \{0,1\} \, \forall i \in [n] \Big\} = X^T[0,1]^n$$

- $\mathcal{Z}\left(\begin{bmatrix} x_1^\intercal \\ x_2^\intercal \end{bmatrix}\right) = \mathbf{conv}\{0, x_1, x_2, x_1 + x_2\} \subseteq \mathbb{R}^2$

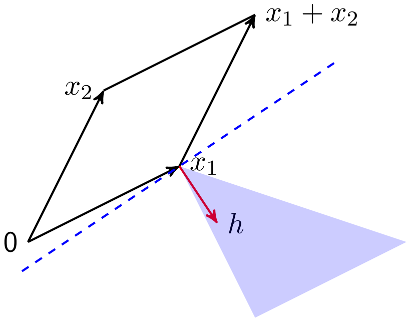


- sample vertices via support functions: $\arg\max_{z \in \mathcal{Z}} h^T z$

Zonotopes

$$\mathcal{Z}(X) := \mathbf{conv}\Big\{ \textstyle\sum_i x_i u_i, \quad u_i \in [0,1]\, \forall i \in [n] \Big\} = X^T[0,1]^n$$

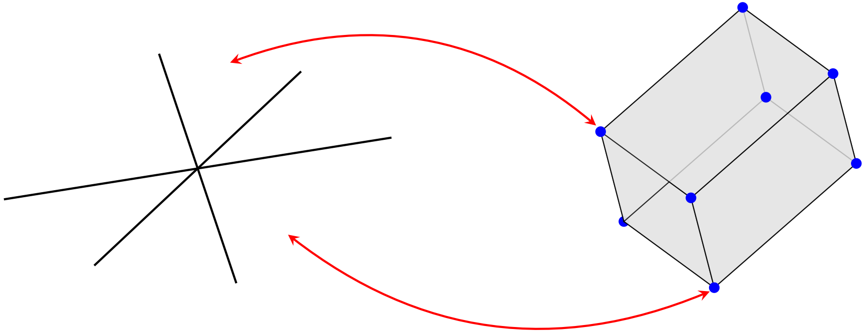
- $\mathcal{Z}\left(\begin{bmatrix} x_1^\intercal \\ x_2^\intercal \end{bmatrix}\right) = \mathbf{conv}\{0, x_1, x_2, x_1 + x_2\} \subseteq \mathbb{R}^2$



- sample vertices via support functions: $\arg\max_{z \in \mathcal{Z}} h^T z$
- $h \in$ **normal cone at** $z \iff$ vertex z is sampled

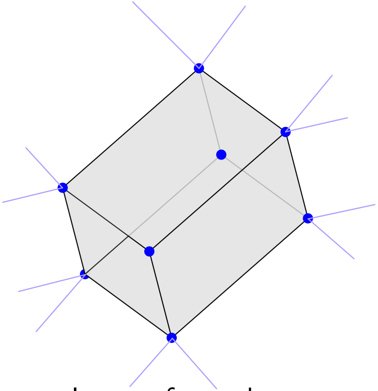
Zonotope-quantized embedding duality

- $\max_{u_i \in [0,1]\, \forall i \in [n]} h^T \sum_i x_i u_i = \sum_i (x_i^T h)_+$ where $u_i^* = 1[x_i^T h \geq 0]$



- a random vector h maps to a vertex of the zonotope via $1[Xh \geq 0] = \arg\max_{z \in \mathcal{Z}} h^T z$
- chambers of the hyperplane arrangements of X correspond to vertices of $\mathcal{Z}(X)$

[Fukuda 2004; Stinson & Gleich & Constantine 2016]



- normalized solid angles of normal cones for each vertex v_i are

$$\theta_i = \#\text{vertices} \cdot \mathbb{P}_{h \sim \mathcal{N}(0,I)}\big[v_i = 1[Xh \geq 0]\big]$$

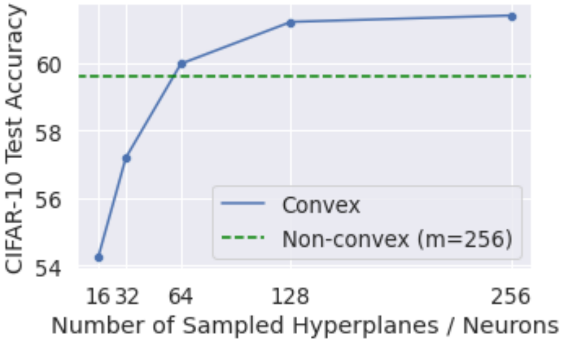
minimum angle $\theta := \min_i \theta_i > 0$ controls the hardness of sampling

Reducing Complexity: Approximating Convex Programs by Sampling

- sampled convex model: sample $D_1, ..., D_{\tilde{p}}$ as $\mathrm{Diag}(Xh_i \geq 0)$ where $h_i \sim N(0, I)$
- **Theorem:** For any integer $k \in \{1, ..., d\}$, we obtain $(1 + \frac{\sigma_{k+1}(X)}{\lambda})$-factor approximation using $O\big(\theta^{-1}(n/k)^k \log(n/k)\big)$ samples. Here, θ is the minimum solid angle of $\mathcal{Z}(X_k)$ where X_k is the best rank-k approximation of X and $\sigma_k = \sigma_k(X)$.

Reducing Complexity: Approximating Convex Programs by Sampling

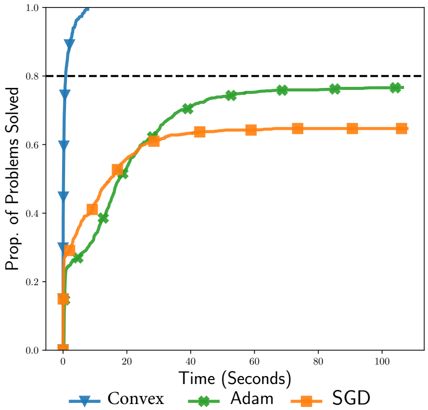
- sampled convex model: sample $D_1, ..., D_{\tilde{p}}$ as $\text{Diag}(Xh_i \geq 0)$ where $h_i \sim N(0, I)$
- **Theorem:** For any integer $k \in \{1, ..., d\}$, we obtain $(1 + \frac{\sigma_{k+1}(X)}{\lambda})$-factor approximation using $O\big(\theta^{-1}(n/k)^k \log(n/k)\big)$ samples. Here, θ is the minimum solid angle of $\mathcal{Z}(X_k)$ where X_k is the best rank-k approximation of X and $\sigma_k = \sigma_k(X)$.



Specialized Convex Solver: Performance Profile

- **baseline:** gradient based non-convex optimization: SGD, ADAM (best of 10 random initializations and 10 learning rates)
- **convex:** proximal gradient with adaptive acceleration

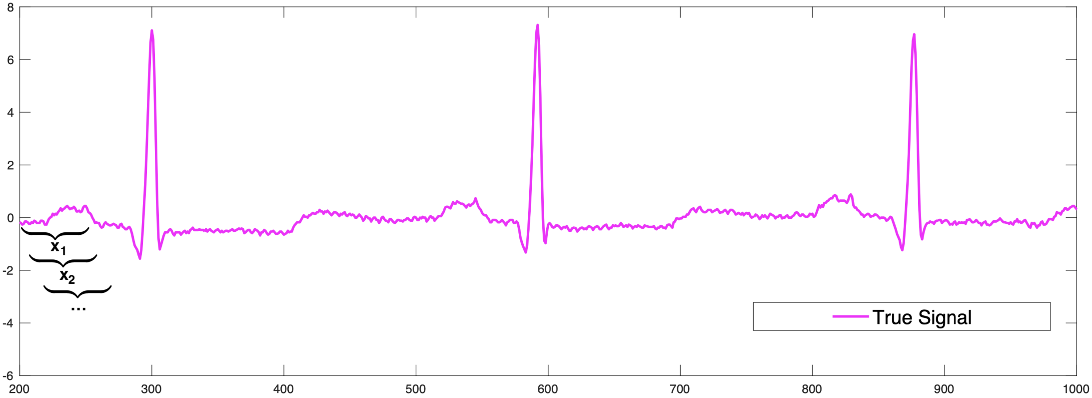
  $O(1/T^2)$ convergence rate



Performance profile showing the pecentage of problems solved over a collection of 400 UC Irvine datasets up to 10^{-3} training error vs time

[Mishkin & Sahiner & Pilanci, ICML 2022] `github.com/pilancilab/scnn`

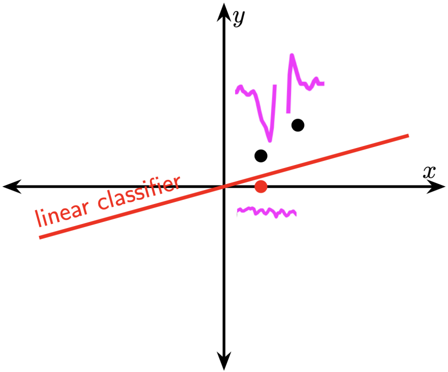
Interpreting Neural Networks via Sketching: Time Series Prediction



$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} = \begin{bmatrix} x[1] & \dots & x[d] \\ x[2] & \dots & x[d+1] \\ \vdots & & \\ x[n] & \dots & x[d+n-1] \end{bmatrix}, \quad y = \begin{bmatrix} x[d+1] \\ x[d+2] \\ \vdots \\ x[d+n] \end{bmatrix}$$

Interpreting Neural Networks via Sketching: Time Series Prediction

$$p_{\text{convex}} = \text{minimize}_{u_1,v_1\ldots u_p,v_p\in\mathcal{K}} \left\| \sum_{i=1}^{p} D_i X(u_i - v_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|u_i\|_2 + \|v_i\|_2 \right)$$



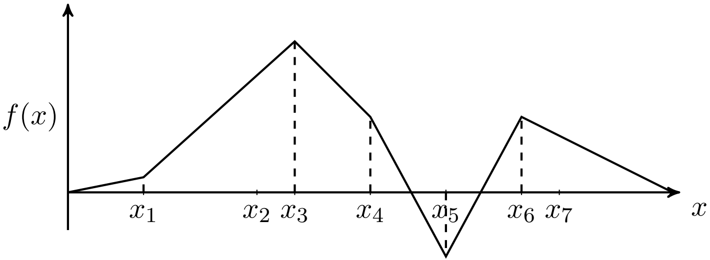
- sampled convex program: $D_i = \text{diag}(Xh_i \geq 0)$, $h_i \sim \mathcal{N}(0, I)$ forms a locality sensitive hash of the data (i.e., a quantized embedding)

Outline

Simplification: Neural networks for one-dimensional data are Lasso models

- one-dimensional data $x_i, y_i \in \mathbb{R}$, $i = 1, ..., n$ ◦ model $f(x) = \sum_{j=1}^{m} \sigma(x w_j^{(1)} + b_j) w_j^{(2)}$

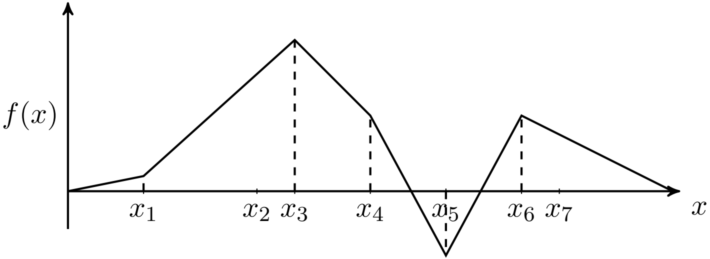
$$p_{\text{non-convex}} = p_{\text{convex}} = \min_{\alpha, b} \|K\alpha + 1b - y\|_2^2 + \lambda \|\alpha\|_1$$ where $K_{ij} = (x_i - x_j)_+ \, \forall i, j \in [n]$



Simplification: Neural networks for one-dimensional data are Lasso models

- one-dimensional data $x_i, y_i \in \mathbb{R}$, $i = 1, ..., n$
- model $f(x) = \sum_{j=1}^{m} \sigma(x w_j^{(1)} + b_j) w_j^{(2)}$

$$p_{\text{non-convex}} = p_{\text{convex}} = \min_{\alpha, b} \|K\alpha + 1b - y\|_2^2 + \lambda\|\alpha\|_1 \qquad \text{where } K_{ij} = (x_i - x_j)_+ \, \forall i, j \in [n]$$



- $K_{ij} = (x_i - x_j)_+$

  $= \mathbf{Vol}([x_i, x_j])_+$ **positive part of the signed volume of** $[x_i, x_j]$

Two-dimensional data

- Suppose data $x_i \in \mathbb{R}^2, y_i \in \mathbb{R}$

$$p_{\text{convex}} = \min_{\alpha, b} \|K\alpha + 1b - y\|_2^2 + \lambda\|\alpha\|_1$$

- $K_{ij} = \frac{2}{\|x_j\|_2}\mathbf{Vol}(\triangle(0, x_i, x_j))_+$
  where $\triangle(a, b, c)$ is the triangle with vertices a, b, c

[Pilanci, From Complexity to Clarity: Analytical Expressions of Deep Neural Network Weights via Clifford's Geometric Algebra and Convexity, 2023]

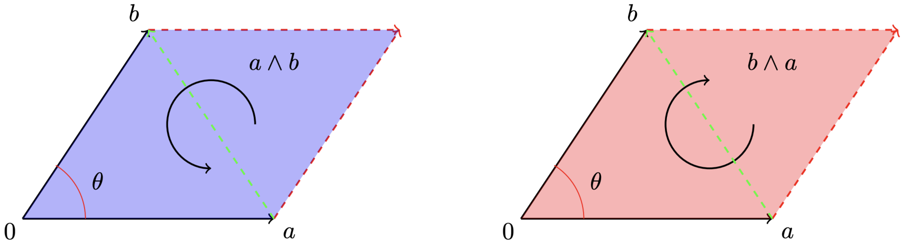
Neural networks are approximately Lasso models

$$p_{\text{convex}} = \min_{\alpha,b} \|K\alpha + 1b - y\|_2^2 + \lambda\|\alpha\|_1$$

- $K_{ij} = \|\times\,(x_{j_1}, ..., x_{j_d})\|_2^{-1}\mathbf{Vol}\big(\mathcal{P}(x_i, x_{j_1}, ..., x_{j_d})\big) \in \mathbb{R}^{n\times\binom{n}{d}}$
  where $\mathcal{P}(a_1, ..., a_k)$ stands for the parallelogram spanned by $a_1, ..., a_k$ and $j = (j_1, ..., j_d)$ is a multi-index
- solutions of p_{convex} are $(1+\epsilon)$-optimal if X is an $\ell_2 \to \ell_1$ subspace embedding, i.e.,
$$(1-\epsilon)\|z\|_2 \leq \frac{1}{n}\|Xz\|_1 \leq (1+\epsilon)\|z\|_2\,\forall z$$
- this condition holds with high probability if X is i.i.d. Gaussian and $n \geq \epsilon^{-4}d$

[Pilanci, From Complexity to Clarity: Analytical Expressions of Deep Neural Network Weights via Clifford's Geometric Algebra and Convexity, 2023]

Clifford (geometric) algebra $\mathbb{G}^d$



- generalizes linear algebra, complex numbers, quaternions, cross-products
- **multivectors:** scalars + vectors + bivectors +...
- the wedge product $a \wedge b$ is a bivector representing the oriented area spanned by a and b
- **geometric product of vectors:** $ab = a \cdot b + a \wedge b$ produces a multivector
- there exists a multiplicative inverse
- duals of multivectors represent complementary subspaces, e.g., $\star e_1 = e_2$ in $\mathbb{G}^2$

Clifford algebra

$$\min_{\alpha,b} \|K\alpha + 1b - y\|_2^2 + \lambda\|\alpha\|_1$$

- $K_{ij} = \frac{(x_i \wedge x_{j_1}, \ldots \wedge x_{j_{d-1}})_+}{\|x_{j_1} \wedge \ldots \wedge x_{j_{d-1}}\|_2} = \mathbf{dist}(x_i, \mathbf{Affine}(x_{j_1,\ldots,x_{j_d}}))$
- optimal neurons are scalar multiples of the duals of $x_{j_1} \wedge \ldots \wedge x_{j_{d-1}}$

Clifford algebra

$$\min_{\alpha,b} \|K\alpha + 1b - y\|_2^2 + \lambda\|\alpha\|_1$$

- $K_{ij} = \frac{(x_i \wedge x_{j_1}, \ldots \wedge x_{j_{d-1}})_+}{\|x_{j_1} \wedge \ldots \wedge x_{j_{d-1}}\|_2} = \mathbf{dist}(x_i, \mathbf{Affine}(x_{j_1, \ldots, x_{j_d}}))$
- optimal neurons are scalar multiples of the duals of $x_{j_1} \wedge \ldots \wedge x_{j_{d-1}}$
- **sketching in Clifford Algebra**

  quantized embedding $1[Xh \geq 0]$ subsamples the indices $(j_1, \ldots, j_d)$

  alternative scheme: sketching data $X \to XS$ preserves distances via JL embeddings

Conclusion

- neural networks are high-dimensional convex models
- better algorithms through randomized dimension reduction

open problems:

- designing better sampling strategies for the convex program
- exploring sketching in Clifford algebra in a unified way

Ref 1 M. Pilanci, From Complexity to Clarity: Analytical Expressions of Deep Neural Network Weights via Clifford's Geometric Algebra and Convexity arXiv, 2023

Ref 2 M. Pilanci, T. Ergen, Path Regularization..., Neurips 2023

Ref 3 M. Pilanci, T. Ergen, Neural Networks are Convex Regularizers..., ICML 2020

papers & code: `https://stanford.edu/~pilanci`

extensions

Three layer NN: FC-Relu-FC-Relu-FC is equivalent to a convex program with double hyperplane arrangements

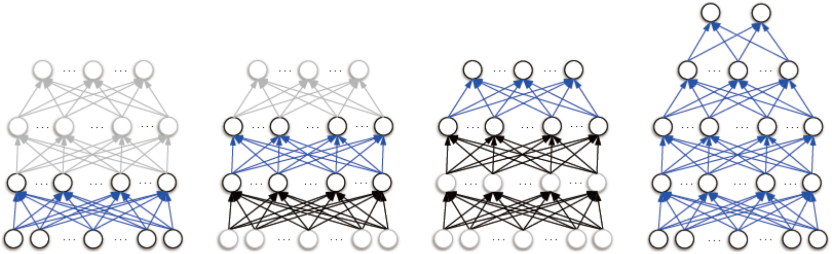
$$p_3^* = \min_{\substack{\{W_j, u_j, w_{1j}, w_{2j}\}_{j=1}^m \\ u_j \in \mathcal{B}_2, \forall j}} \frac{1}{2} \left\| \sum_{j=1}^m \left((\mathbf{X} W_j)_+ w_{1j} \right)_+ w_{2j} - y \right\|_2^2 + \frac{\beta}{2} \sum_{j=1}^m \left(\|W_j\|_F^2 + \|w_{1j}\|_2^2 + w_{2j}^2 \right),$$

Theorem

The equivalent convex problem is

$$\min_{\{W_i, W_i'\}_{i=1}^p \in \mathcal{K}} \frac{1}{2} \left\| \sum_{i=1}^p \sum_{j=1}^P D_i D_j \tilde{\mathbf{X}} \left(W_{ij}' - W_{ij} \right) - y \right\|_2^2 + \frac{\beta}{2} \sum_{i,j=1}^p \|W_{ij}\|_F + \|W_{ij}'\|_F$$

Layer-Wise Training of Deep Networks



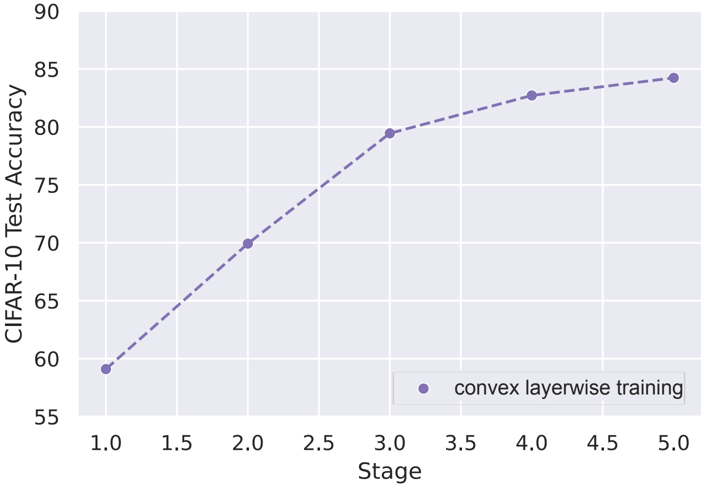
(i) train a two-layer network convex optimization

(ii) fix the hidden layer to use as feature embedding

(ii) repeat two-layer network training on these features

- ideal for **edge AI:** low memory and low communication between blocks
- modular: networks can keep evolving, can terminate early during inference
- each convex model is trained to global optimality efficiently with no hyperparameter tuning

Numerical results for layer-wise convex learning: CIFAR-10 image classification



- end-to-end trained 5 layer CNN accuracy: 89%, 16 layer VGG accuracy: 92%

ReLU Networks with Batch Normalization (BN)

- BN transforms a batch of data to zero mean and standard deviation one, and has two trainable parameters α, γ

$$\mathbf{BN}_{\alpha,\gamma}(x) = \frac{(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)x}{\|(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)x\|_2}\gamma + \alpha$$

$$p_{\text{non-convex}} = \min_{W_1, W_2, \alpha, \gamma} \left\|\mathbf{BN}_{\alpha,\gamma}(\phi(XW_1))W_2 - y\right\|_2^2 + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

$$\|$$

$$p_{\text{convex}} = \min_{w_1, v_1 \ldots w_p, v_p \in \mathcal{K}} \left\|\sum_{i=1}^p U_i(w_i - v_i) - y\right\|_2^2 + \lambda\left(\sum_{i=1}^p \|w_i\|_2 + \|v_i\|_2\right)$$

where $U_i \Sigma_i V_i^T = D_i X$ is the SVD of DX_i, i.e., BatchNorm whitens local data

T. Ergen, A. Sahiner, B. Ozturkler, J. Pauly, M. Mardani, M. Pilanci **Demystifying Batch Normalization in ReLU Networks, ICLR 2022**

Vector Output Two-layer ReLU: equivalent to nuclear norm penalty

$$p_{\text{non-convex}} = \min_{W_1 \in \mathbb{R}^{d\times m}, W_2 \in \mathbb{R}^{m\times c}} \left\| \sum_{j=1}^{m} \phi(XW_{1j})W_{2j} - Y \right\|_2^2 + \lambda \left(\|W_1\|_F^2 + \|W_2\|_F^2 \right)$$

$$p_{\text{convex}} = \min_{U_1, V_1 \ldots U_p, V_p \in \mathcal{K}} \left\| \sum_{i=1}^{p} D_i X (U_i - V_i) - y \right\|_2^2 + \lambda \left(\sum_{i=1}^{p} \|U_i\|_* + \|V_i\|_* \right)$$

$D_1, ..., D_p$ are fixed diagonal matrices

Theorem $p_{\text{non-convex}} = p_{\text{convex}}$, and an optimal solution to $p_{\text{non-convex}}$ can be recovered from optimal non-zero U_i^*, V_i^*, $i = 1, ..., p$.

A. Sahiner, T. Ergen, J. Pauly, M. Pilanci **Vector-output ReLU Neural Network Problems are Copositive Programs, ICLR 2021**

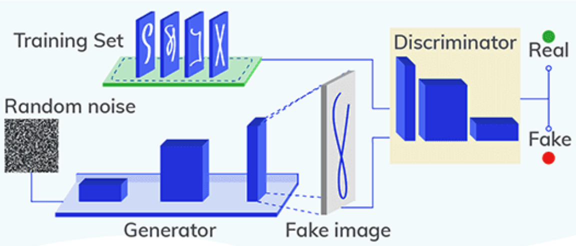
All stationary points correspond to sampled convex models

$$p_{\text{non-convex}} := \ \text{minimize}_{W_1,W_2} \quad L\left(\phi(XW_1)W_2, y\right) + \lambda\left(\|W_1\|_F^2 + \|W_2\|_F^2\right)$$

Theorem Stationary points $\left\{x \ : 0 \in \operatorname{conv}\left\{\lim_{k\to\infty} \nabla f(x_k) \mid \lim_{k\to\infty} x_k = x,\, x_k \in D\right\}\right\}$ of $p_{\text{non-convex}}$ are optimal solutions of the sampled convex program $p_{\text{sampled-cvx}}$

Y. Wang, J. Lacotte, M. Pilanci. **The Hidden Convex Optimization Landscape of Two-Layer ReLU Neural Networks: an Exact Characterization of the Optimal Solutions ICLR, 2022**

Convex Generative Adversarial Networks (GANs)



- Wasserstein GAN parameterized with neural networks

$$p^* = \min_{\theta_g} \max_{D:\ \text{1-Lipschitz}} \mathbb{E}_{x\sim p_x}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G_{\theta_g}(z))]$$

$$\cong \min_{\theta_g} \max_{\theta_d} \mathbb{E}_{x\sim p_x}[D_{\theta_d}(x)] - \mathbb{E}_{z\sim p_z}[D_{\theta_d}(G_{\theta_g}(z))]$$

Theorem: Two-layer generator two-layer discriminator WGAN problems are convex-concave games. Saddle-points exists and globally solvable. (Sahiner et al. **Hidden Convexity of Wasserstein GANs, ICLR 2022.)**

Transformer and Attention-based Architectures

- based on the attention module

$$f(X) = \sigma(XQ^T KX)XV$$

- Q, K, V are trainable parameters: Q : query, K : key, V : value
- used in transformers, vision transformers, mixer models...
- **There is a convex formulation[1], which involves the nuclear norm**

[1] A. Sahiner, T. Ergen, B. Ozturkler, M. Mardani, J. Pauly, M. Pilanci, ICML 2022

Exact Convex Program: Two-Layer ReLU NN

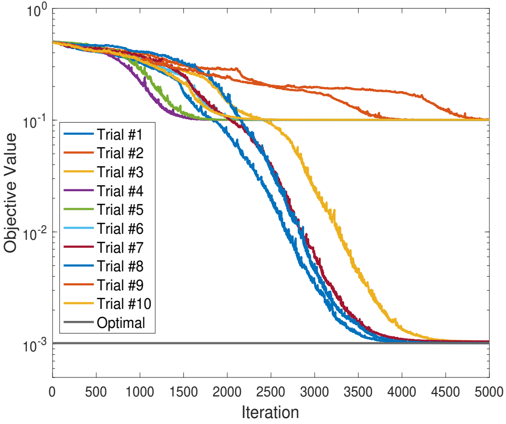


Figure: $m = 8$

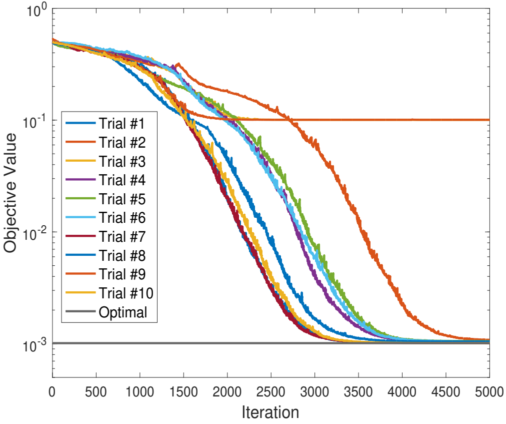
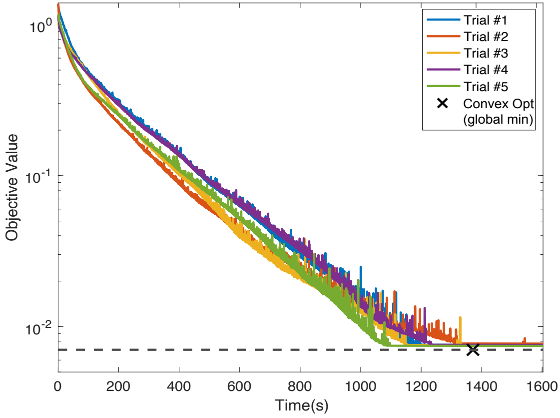


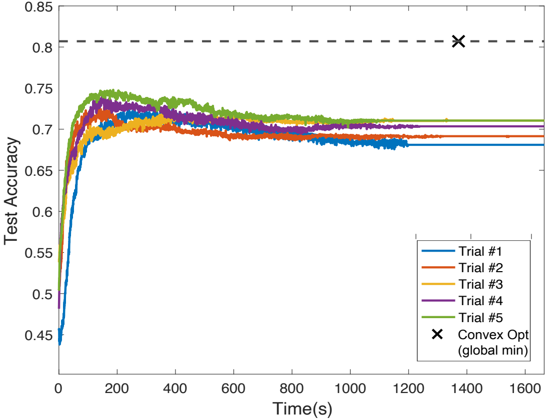
Figure: $m = 15$

Training cost of a two-layer ReLU network trained with SGD (10 initialization trials) and the convex program on a toy dataset ($d = 2$)

Exact Convex Program: Classifying a subset of CIFAR-10



(a) $m = 8$



(b) $m = 50$

Figure: Two-layer ReLU network trained with SGD (10 initialization trials) and the convex program on

Sampled Convex Model vs Non-convex Model (Stochastic Gradient Descent)

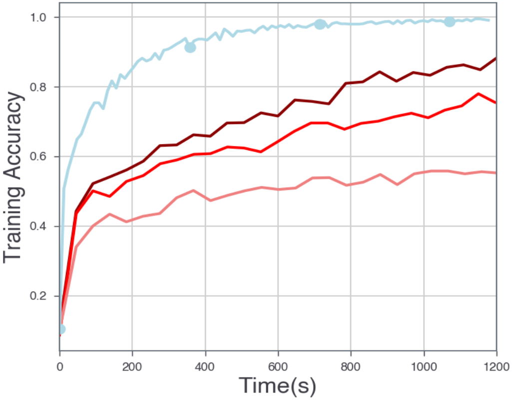


Figure: training accuracy

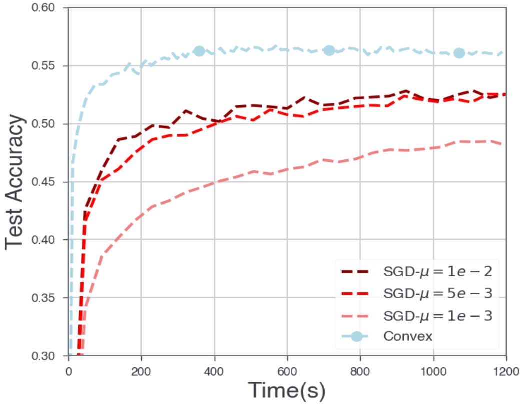
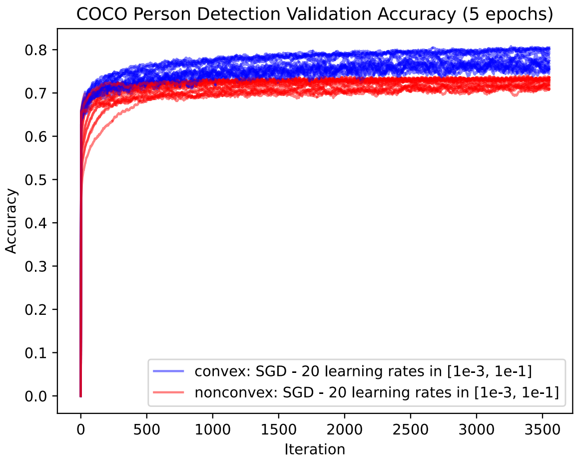


Figure: test accuracy

10-class classification on the CIFAR Dataset ($n = 50,000$, $d = 3072$) with randomly sampled

Re-training Final Convolutional Layers of Pretrained Deep Nets



Person detection task on the COCO Dataset containing $110,000$ images of median resolution 640 x 480. Two-layer ReLU CNN trained on pretrained MobileNetV3 features (convex