Interpretable Global Minima of Deep ReLU Neural Networks on Sequentially Separable Data

Thomas Chen and Patrícia Muñoz Ewald

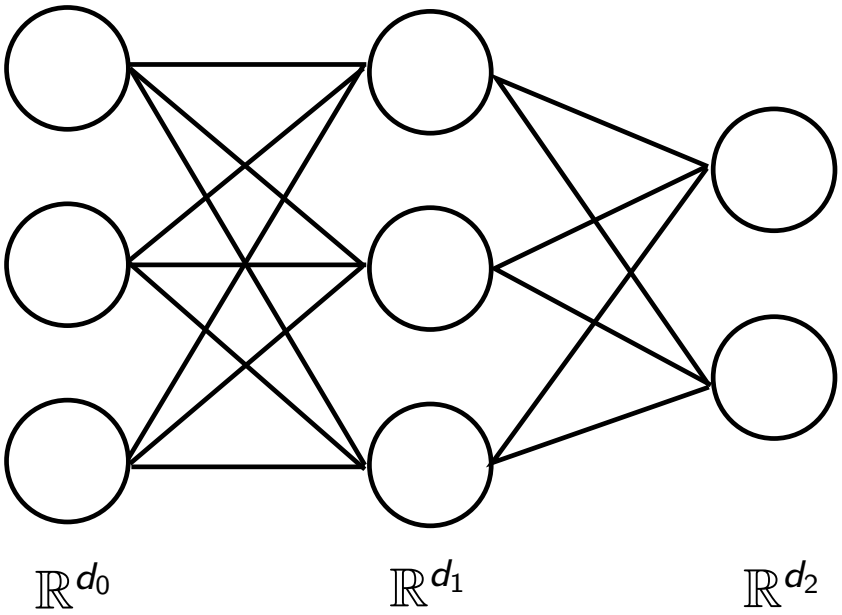
Dept. of Mathematics, University of Texas at Austin

Setup

Consider a neural network

$$\begin{aligned} x^{(0)} &= x_0 \in \mathbb{R}^{d_0} \quad \text{initial input,} \\ x^{(\ell)} &= \sigma(W_\ell x^{(\ell-1)} + b_\ell) \in \mathbb{R}^{d_\ell}, \ \text{for } \ell = 1, \cdots, L-1, \\ x^{(L)} &= W_L x^{(L-1)} + b_L \in \mathbb{R}^{d_L}. \end{aligned} \tag{3.1}$$

Usually, will take $\sigma(x)_i = \max\{0, x_i\}$ (ReLU).



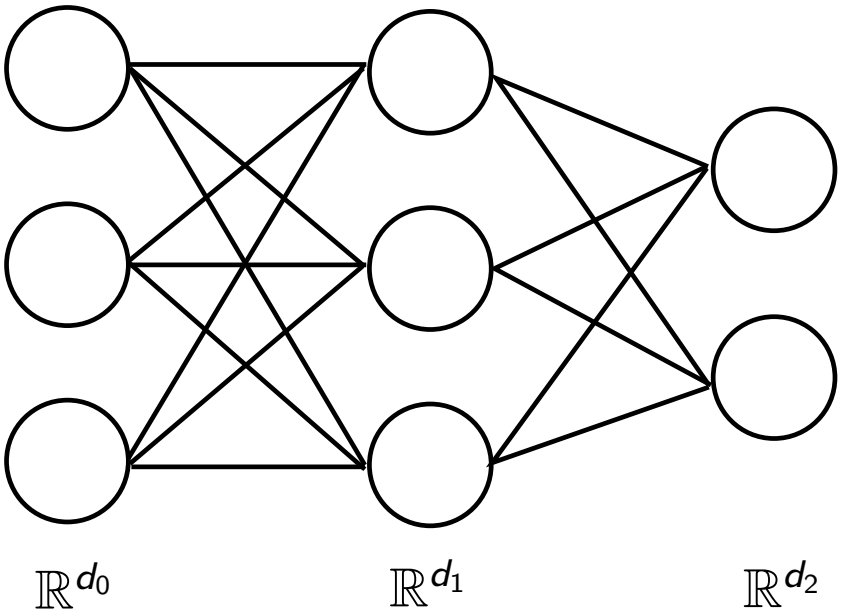
Goal: Directly compare features or representations of data in different layers.

Setup

Consider a neural network

$$\begin{aligned} x^{(0)} &= x_0 \in \mathbb{R}^{d_0} \quad \text{initial input,} \\ x^{(\ell)} &= \sigma(W_\ell x^{(\ell-1)} + b_\ell) \in \mathbb{R}^{d_\ell}, \ \text{for } \ell = 1, \cdots, L-1, \\ x^{(L)} &= W_L x^{(L-1)} + b_L \in \mathbb{R}^{d_L}. \end{aligned} \tag{3.1}$$

Usually, will take $\sigma(x)_i = \max\{0, x_i\}$ (ReLU).



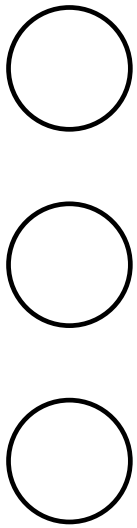
Goal: Directly compare features or representations of data in different layers.

Setup

Consider a neural network

$$\begin{aligned} x^{(0)} &= x_0 \in \mathbb{R}^{d_0} \quad \text{initial input,} \\ x^{(\ell)} &= \sigma(W_\ell x^{(\ell-1)} + b_\ell) \in \mathbb{R}^{d_\ell}, \ \text{for } \ell = 1, \cdots, L-1, \\ x^{(L)} &= W_L x^{(L-1)} + b_L \in \mathbb{R}^{d_L}. \end{aligned} \tag{3.2}$$

Usually, will take $\sigma(x)_i = \max\{0, x_i\}$ (ReLU).



$\mathbb{R}^{d_0}$

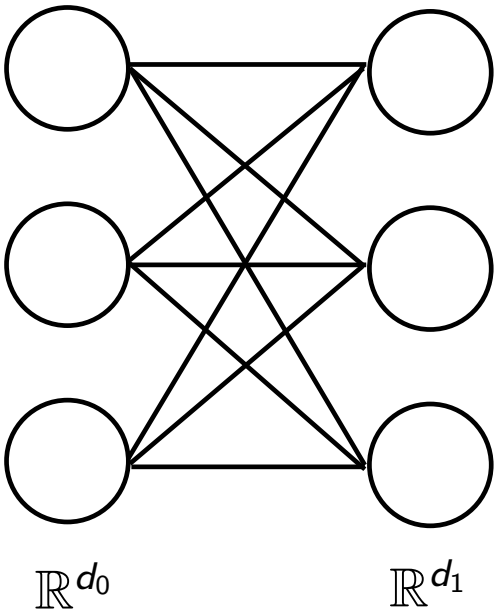
Goal: Directly compare features or representations of data in different layers.

Setup

Consider a neural network

$$\begin{aligned} x^{(0)} &= x_0 \in \mathbb{R}^{d_0} \quad \text{initial input,} \\ x^{(\ell)} &= \sigma(W_\ell x^{(\ell-1)} + b_\ell) \in \mathbb{R}^{d_\ell}, \ \text{for } \ell = 1, \cdots, L-1, \\ x^{(L)} &= W_L x^{(L-1)} + b_L \in \mathbb{R}^{d_L}. \end{aligned} \tag{3.3}$$

Usually, will take $\sigma(x)_i = \max\{0, x_i\}$ (ReLU).



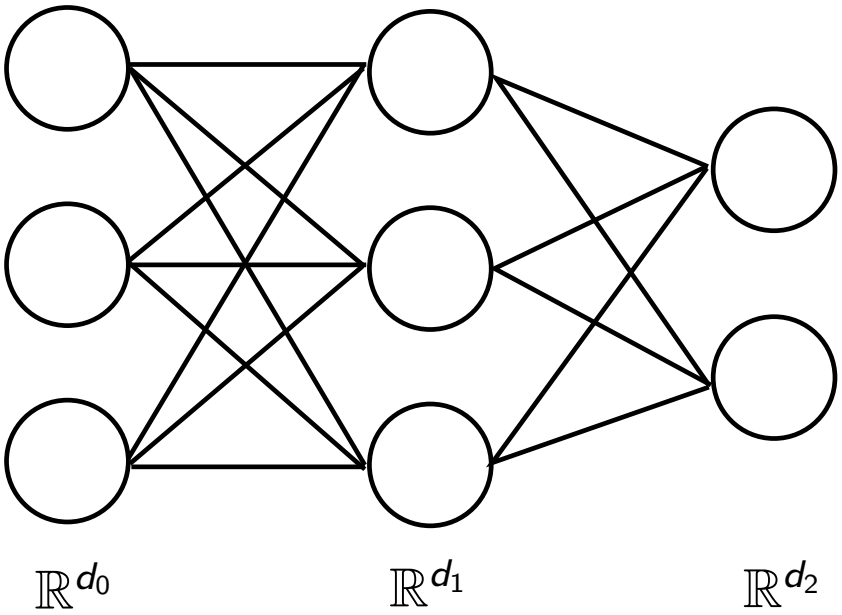
Goal: Directly compare features or representations of data in different layers.

Setup

Consider a neural network

$$\begin{aligned} x^{(0)} &= x_0 \in \mathbb{R}^{d_0} \quad \text{initial input,} \\ x^{(\ell)} &= \sigma(W_\ell x^{(\ell-1)} + b_\ell) \in \mathbb{R}^{d_\ell}, \ \text{for } \ell = 1, \cdots, L-1, \\ x^{(L)} &= W_L x^{(L-1)} + b_L \in \mathbb{R}^{d_L}. \end{aligned} \tag{3.4}$$

Usually, will take $\sigma(x)_i = \max\{0, x_i\}$ (ReLU).



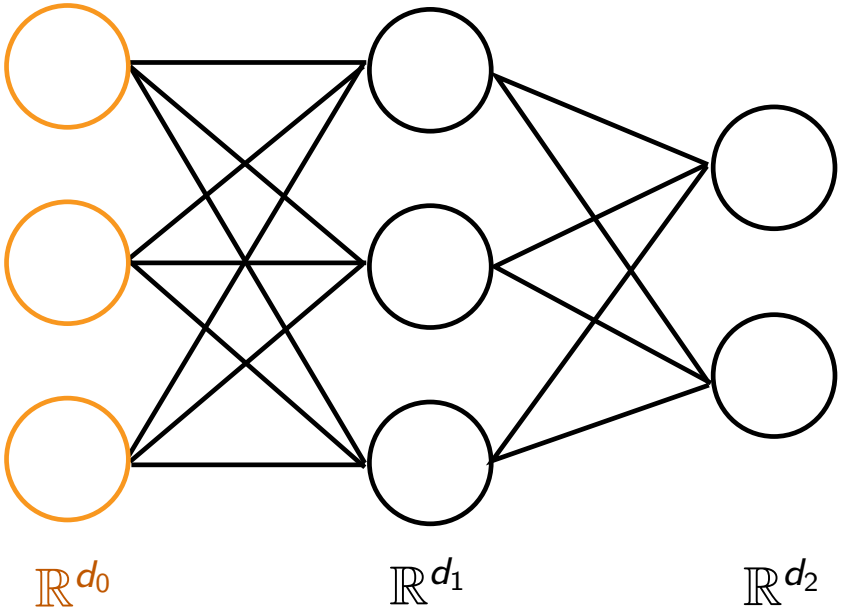
Goal: Directly compare features or representations of data in different layers.

Setup

Consider a neural network

$$\begin{aligned} x^{(0)} &= x_0 \in \mathbb{R}^{d_0} \quad \text{initial input,} \\ x^{(\ell)} &= \sigma(W_\ell x^{(\ell-1)} + b_\ell) \in \mathbb{R}^{d_\ell}, \ \text{for } \ell = 1, \cdots, L-1, \\ x^{(L)} &= W_L x^{(L-1)} + b_L \in \mathbb{R}^{d_L}. \end{aligned} \tag{3.5}$$

Usually, will take $\sigma(x)_i = \max\{0, x_i\}$ (ReLU).

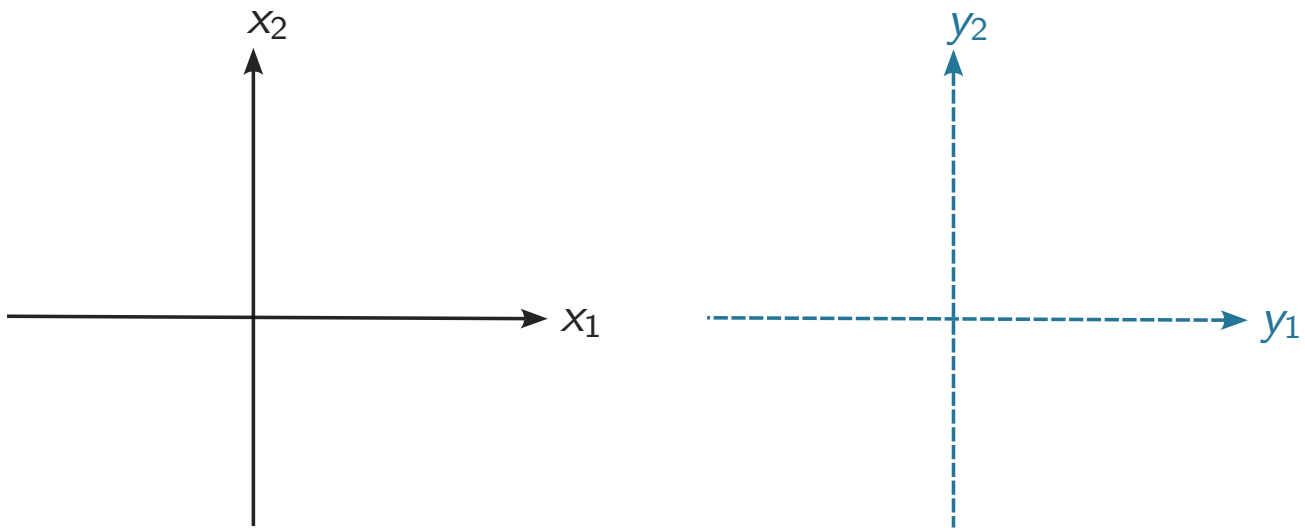


Goal: Directly compare features or representations of data in different layers.

Definition (Chen–E. '23)

Given $W \in \mathbb{R}^{d_1 \times d_0}$ and $b \in \mathbb{R}^{d_1}$ we define the **truncation map**

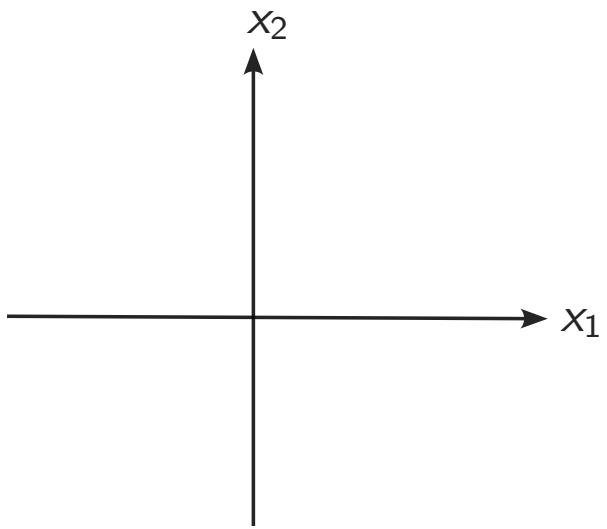
$$\tau_{W,b} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_0},$$
$$x \mapsto W^+ (\sigma(Wx + b) - b).$$

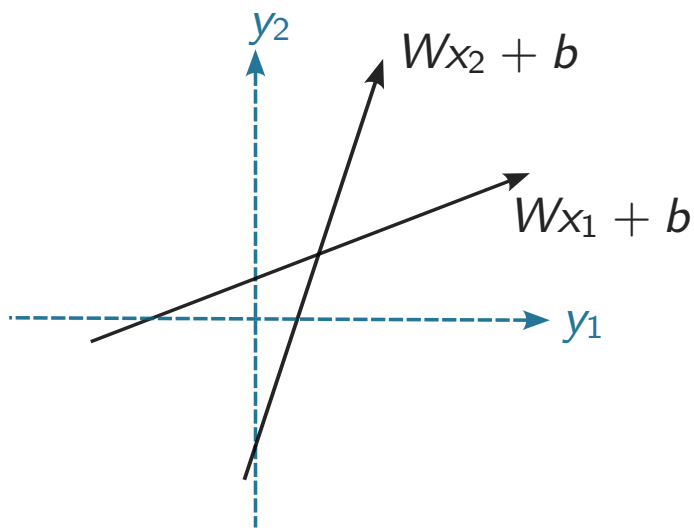


Definition (Chen–E. '23)

Given $W \in \mathbb{R}^{d_1 \times d_0}$ and $b \in \mathbb{R}^{d_1}$ we define the **truncation map**

$$\tau_{W,b} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_0},$$
$$x \mapsto W^+ (\sigma(Wx + b) - b).$$

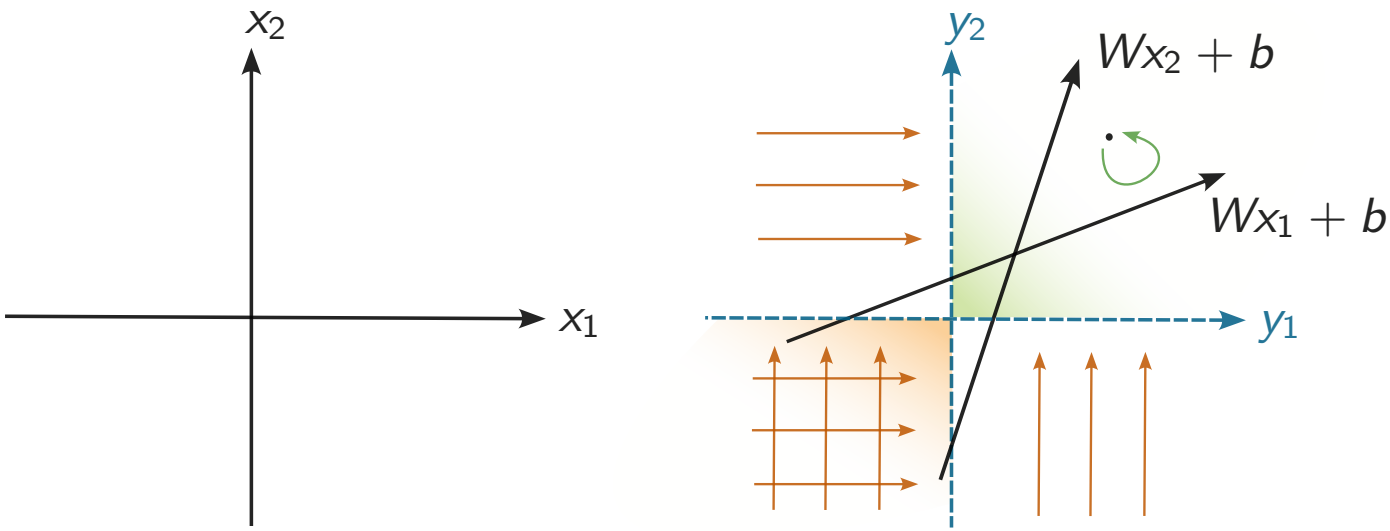




Definition (Chen–E. '23)

Given $W \in \mathbb{R}^{d_1 \times d_0}$ and $b \in \mathbb{R}^{d_1}$ we define the **truncation map**

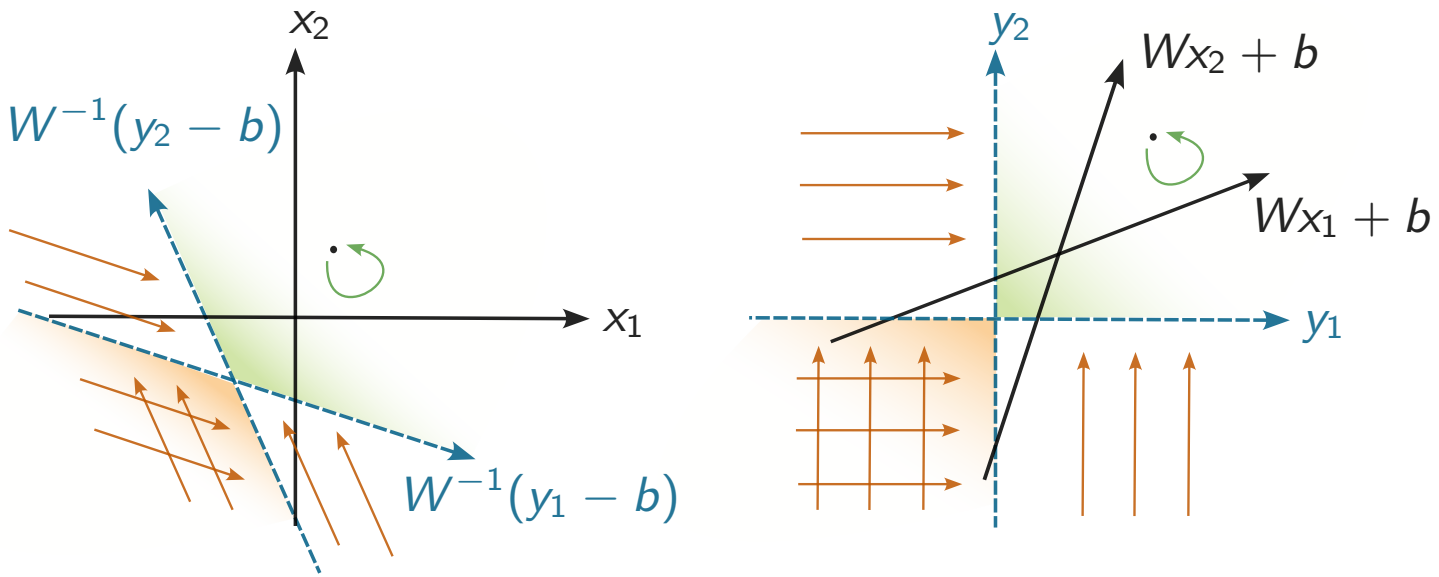
$$\tau_{W,b} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_0},$$
$$x \mapsto W^+ \left(\sigma(Wx + b) - b\right).$$



Definition (Chen–E. '23)

Given $W \in \mathbb{R}^{d_1 \times d_0}$ and $b \in \mathbb{R}^{d_1}$ we define the **truncation map**

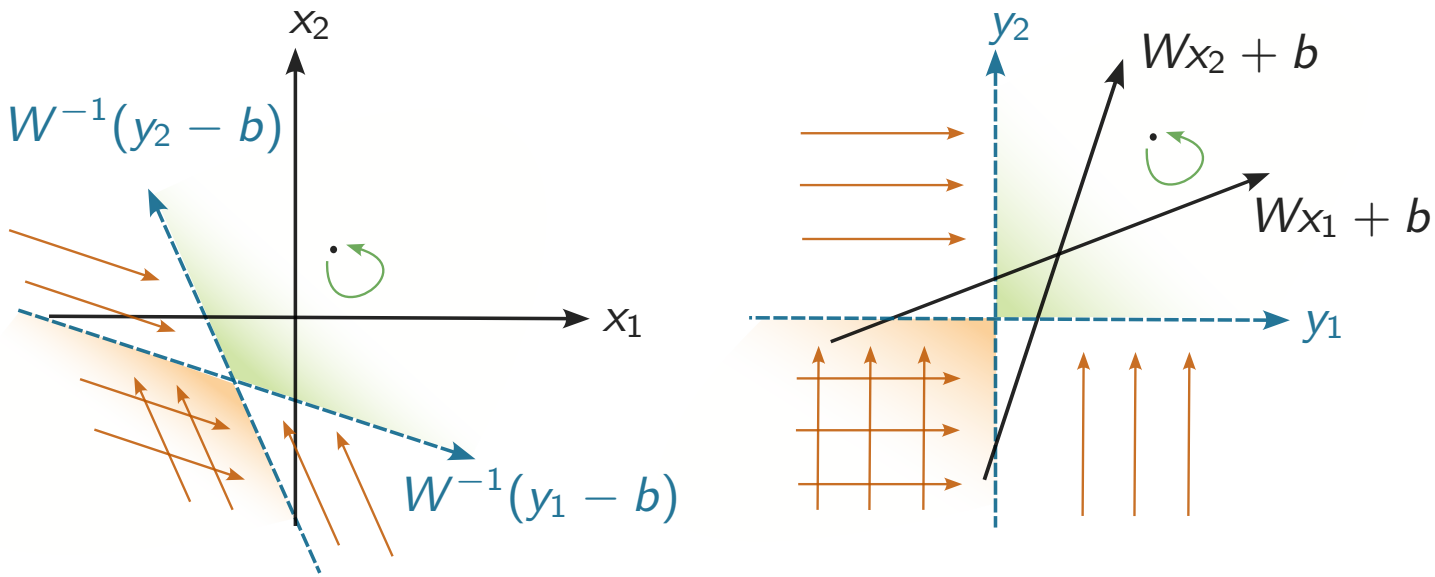
$$\tau_{W,b} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_0},$$
$$x \mapsto W^{+}\left(\sigma(Wx + b) - b\right).$$



Definition (Chen–E. '23)

Given $W \in \mathbb{R}^{d_1 \times d_0}$ and $b \in \mathbb{R}^{d_1}$ we define the **truncation map**

$$\tau_{W,b} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_0},$$
$$x \mapsto W^{+}\left(\sigma(Wx + b) - b\right).$$



Truncation map: What about more layers?

Define **cumulative parameters**

$$W^{(1)} := W_1, \quad W^{(\ell)} := W_\ell \cdots W_1 = W_\ell W^{(\ell-1)},$$
$$b^{(1)} := b_1, \quad b^{(\ell)} := W_\ell b^{(\ell-1)} + b_\ell, \tag{3.6}$$

and

$$x^{(\tau,0)} := x^{(0)}, \quad x^{(\tau,\ell)} := \tau_{W^{(\ell)},b^{(\ell)}}(x^{(\tau,\ell-1)}). \tag{3.7}$$

Proposition (Chen–E. '25)

If all weight matrices W_ℓ are surjective, $\ell = 1, \cdots, L$, then

$$x^{(\ell)} = W^{(\ell)} x^{(\tau,\ell)} + b^{(\ell)}, \tag{3.8}$$

for $\ell = 1, \cdots, L-1$, and

$$x^{(L)} = W^{(L)} x^{(\tau,L-1)} + b^{(L)}. \tag{3.9}$$

Truncation map: What about more layers?

Define **cumulative parameters**

$$W^{(1)} := W_1, \quad W^{(\ell)} := W_\ell \cdots W_1 = W_\ell W^{(\ell-1)},$$
$$b^{(1)} := b_1, \quad b^{(\ell)} := W_\ell b^{(\ell-1)} + b_\ell, \tag{3.6}$$

and

$$x^{(\tau,0)} := x^{(0)}, \quad x^{(\tau,\ell)} := \tau_{W^{(\ell)},b^{(\ell)}}(x^{(\tau,\ell-1)}). \tag{3.7}$$

Proposition (Chen–E. '25)

If all weight matrices W_ℓ are surjective, $\ell = 1, \cdots, L$, then

$$x^{(\ell)} = W^{(\ell)} x^{(\tau,\ell)} + b^{(\ell)}, \tag{3.8}$$

for $\ell = 1, \cdots, L-1$, and

$$x^{(L)} = W^{(L)} x^{(\tau,L-1)} + b^{(L)}. \tag{3.9}$$

Truncation map: What about more layers?

Define **cumulative parameters**

$$W^{(1)} := W_1, \quad W^{(\ell)} := W_\ell \cdots W_1 = W_\ell W^{(\ell-1)},$$
$$b^{(1)} := b_1, \quad b^{(\ell)} := W_\ell b^{(\ell-1)} + b_\ell, \tag{3.6}$$

and

$$x^{(\tau,0)} := x^{(0)}, \quad x^{(\tau,\ell)} := \tau_{W^{(\ell)},b^{(\ell)}}(x^{(\tau,\ell-1)}). \tag{3.7}$$

Proposition (Chen–E. '25)

If all weight matrices W_ℓ are surjective, $\ell = 1, \cdots, L$, then

$$x^{(\ell)} = W^{(\ell)} x^{(\tau,\ell)} + b^{(\ell)}, \tag{3.8}$$

for $\ell = 1, \cdots, L-1$, and

$$x^{(L)} = W^{(L)} x^{(\tau,L-1)} + b^{(L)}. \tag{3.9}$$

Truncation map: What about more layers?

Define **cumulative parameters**

$$W^{(1)} := W_1, \quad W^{(\ell)} := W_\ell \cdots W_1 = W_\ell W^{(\ell-1)},$$
$$b^{(1)} := b_1, \quad b^{(\ell)} := W_\ell b^{(\ell-1)} + b_\ell, \tag{3.6}$$

and

$$x^{(\tau,0)} := x^{(0)}, \quad x^{(\tau,\ell)} := \tau_{W^{(\ell)},b^{(\ell)}}(x^{(\tau,\ell-1)}). \tag{3.7}$$

Proposition (Chen–E. '25)

If all weight matrices W_ℓ are surjective, $\ell = 1, \cdots, L$, then

$$x^{(\ell)} = W^{(\ell)} x^{(\tau,\ell)} + b^{(\ell)}, \tag{3.8}$$

for $\ell = 1, \cdots, L-1$, and

$$x^{(L)} = W^{(L)} x^{(\tau,L-1)} + b^{(L)}. \tag{3.9}$$

What this means

This means that a feedforward neural network

$$f_{\underline{\theta}} : \underbrace{\mathbb{R}^{d_0}}_{\text{Input layer}} \xrightarrow{f_1} \underbrace{\mathbb{R}^{d_1} \xrightarrow{f_2} \cdots \xrightarrow{f_{L-1}} \mathbb{R}^{d_{L-1}}}_{\text{Hidden layers}} \xrightarrow{f_L} \underbrace{\mathbb{R}^{d_L}}_{\text{Output layer}}, \tag{3.10}$$

with decreasing layer dimensions $d_0 \geq d_1 \geq \cdots \geq d_{L-1} \geq d_L$ can be rewritten as a sequence of transformations on input space, followed by an affine map to output space,

$$f_{\underline{\theta}} : \underbrace{\mathbb{R}^{d_0} \xrightarrow{\tau_1} \mathbb{R}^{d_0} \xrightarrow{\tau_2} \cdots \xrightarrow{\tau_{L-1}} \mathbb{R}^{d_0}}_{\text{Truncation maps}} \xrightarrow{\tilde{f}_L} \mathbb{R}^{d_L}. \tag{3.11}$$

This is independent of the activation function(s) being used.

What this means

This means that a feedforward neural network

$$f_{\underline{\theta}} : \underbrace{\mathbb{R}^{d_0}}_{\text{Input layer}} \xrightarrow{f_1} \underbrace{\mathbb{R}^{d_1} \xrightarrow{f_2} \cdots \xrightarrow{f_{L-1}} \mathbb{R}^{d_{L-1}}}_{\text{Hidden layers}} \xrightarrow{f_L} \underbrace{\mathbb{R}^{d_L}}_{\text{Output layer}}, \tag{3.10}$$

with decreasing layer dimensions $d_0 \geq d_1 \geq \cdots \geq d_{L-1} \geq d_L$ can be rewritten as a sequence of transformations on input space, followed by an affine map to output space,

$$f_{\underline{\theta}} : \underbrace{\mathbb{R}^{d_0} \xrightarrow{\tau_1} \mathbb{R}^{d_0} \xrightarrow{\tau_2} \cdots \xrightarrow{\tau_{L-1}} \mathbb{R}^{d_0}}_{\text{Truncation maps}} \xrightarrow{\tilde{f}_L} \mathbb{R}^{d_L}. \tag{3.11}$$

This is independent of the activation function(s) being used.

What this means

This means that a feedforward neural network

$$f_{\underline{\theta}} : \underbrace{\mathbb{R}^{d_0}}_{\text{Input layer}} \xrightarrow{f_1} \underbrace{\mathbb{R}^{d_1} \xrightarrow{f_2} \cdots \xrightarrow{f_{L-1}} \mathbb{R}^{d_{L-1}}}_{\text{Hidden layers}} \xrightarrow{f_L} \underbrace{\mathbb{R}^{d_L}}_{\text{Output layer}}, \tag{3.10}$$

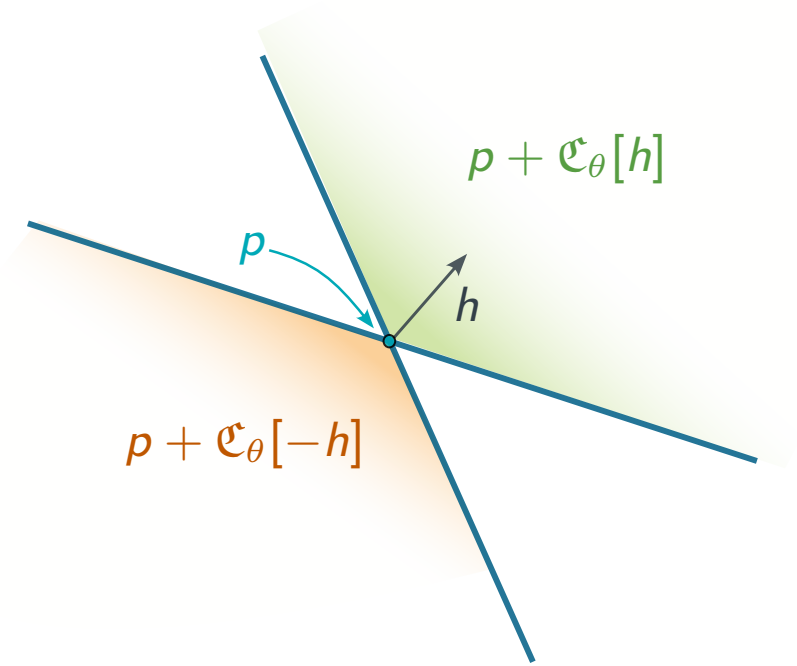
with decreasing layer dimensions $d_0 \geq d_1 \geq \cdots \geq d_{L-1} \geq d_L$ can be rewritten as a sequence of transformations on input space, followed by an affine map to output space,

$$f_{\underline{\theta}} : \underbrace{\mathbb{R}^{d_0} \xrightarrow{\tau_1} \mathbb{R}^{d_0} \xrightarrow{\tau_2} \cdots \xrightarrow{\tau_{L-1}} \mathbb{R}^{d_0}}_{\text{Truncation maps}} \xrightarrow{\tilde{f}_L} \mathbb{R}^{d_L}. \tag{3.11}$$

This is independent of the activation function(s) being used.

Truncation map and (round) cones

Consider the cone $p + \mathfrak{C}_\theta[h] \subseteq \mathbb{R}^n$ centered around a unit vector h and based at p:



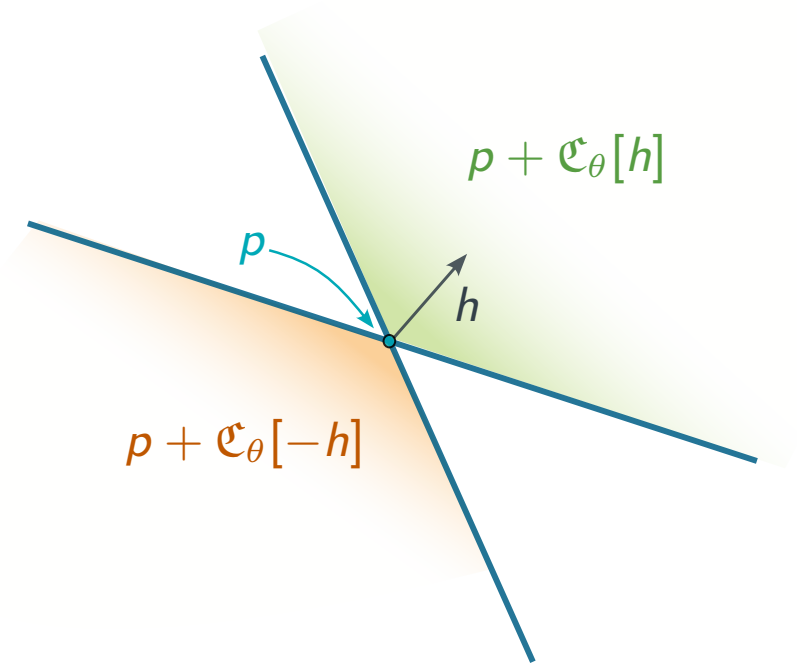
Lemma (Chen–E. '25)

There are W and b such that, for $\sigma = ReLU$,

$$\tau_{W,b}(x) = \begin{cases} x, & x \in p + \mathfrak{C}_\theta[h], \\ p, & x \in p + \mathfrak{C}_\theta[-h]. \end{cases} \tag{4.1}$$

Truncation map and (round) cones

Consider the cone $p + \mathfrak{C}_\theta[h] \subseteq \mathbb{R}^n$ centered around a unit vector h and based at p:



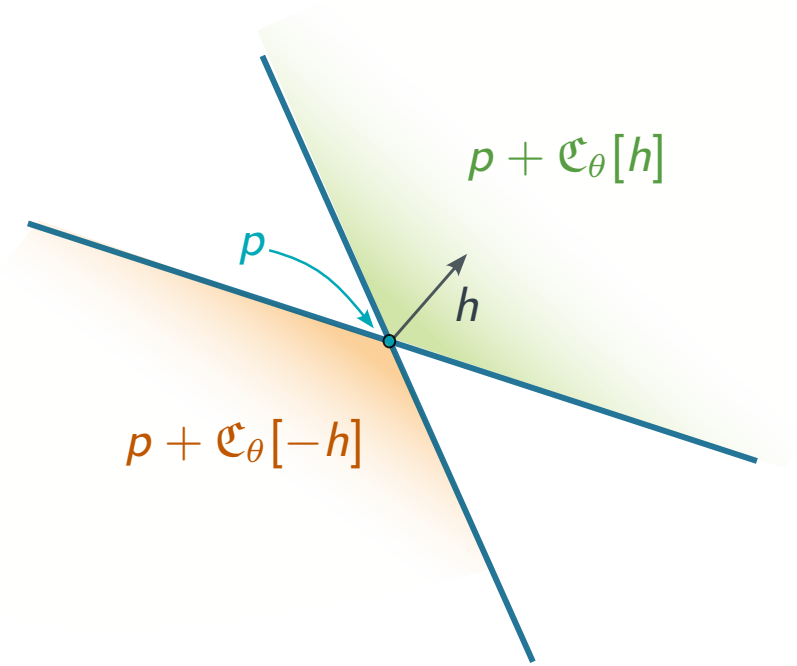
Lemma (Chen–E. '25)

There are W and b such that, for $\sigma = ReLU$,

$$\tau_{W,b}(x) = \begin{cases} x, & x \in p + \mathfrak{C}_\theta[h], \\ p, & x \in p + \mathfrak{C}_\theta[-h]. \end{cases} \tag{4.1}$$

Truncation map and (round) cones

Consider the cone $p + \mathfrak{C}_\theta[h] \subseteq \mathbb{R}^n$ centered around a unit vector h and based at p:



Lemma (Chen–E. '25)

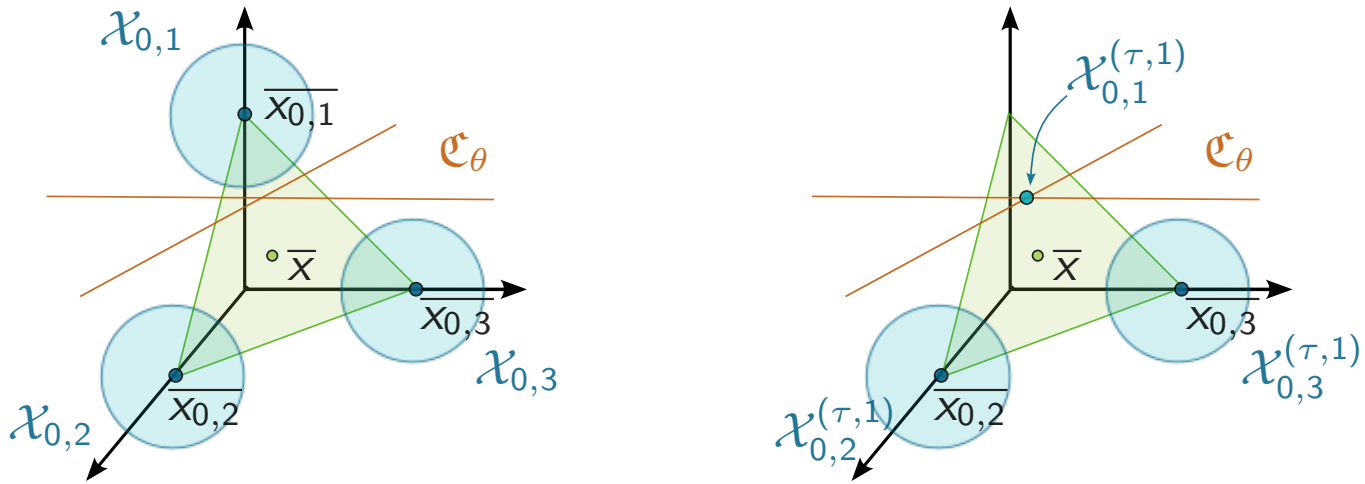
There are W and b such that, for $\sigma = ReLU$,

$$\tau_{W,b}(x) = \begin{cases} W^+ W\, x, & x \in p + \mathfrak{C}_\theta[h], \\ W^+ W\, p, & x \in p + \mathfrak{C}_\theta[-h]. \end{cases} \tag{4.2}$$

Theorem (Chen–E. '25)

Consider a set of training data $\mathcal{X}_0 = \bigcup_{j=1}^{Q} \mathcal{X}_{0,j} \subseteq \mathbb{R}^{d_0}$, with Q classes associated to linearly independent labels $y_j \in \mathbb{R}^Q$. If the data is sufficiently clustered, then there exists a ReLU neural network with $Q+1$ layers which interpolates the data.

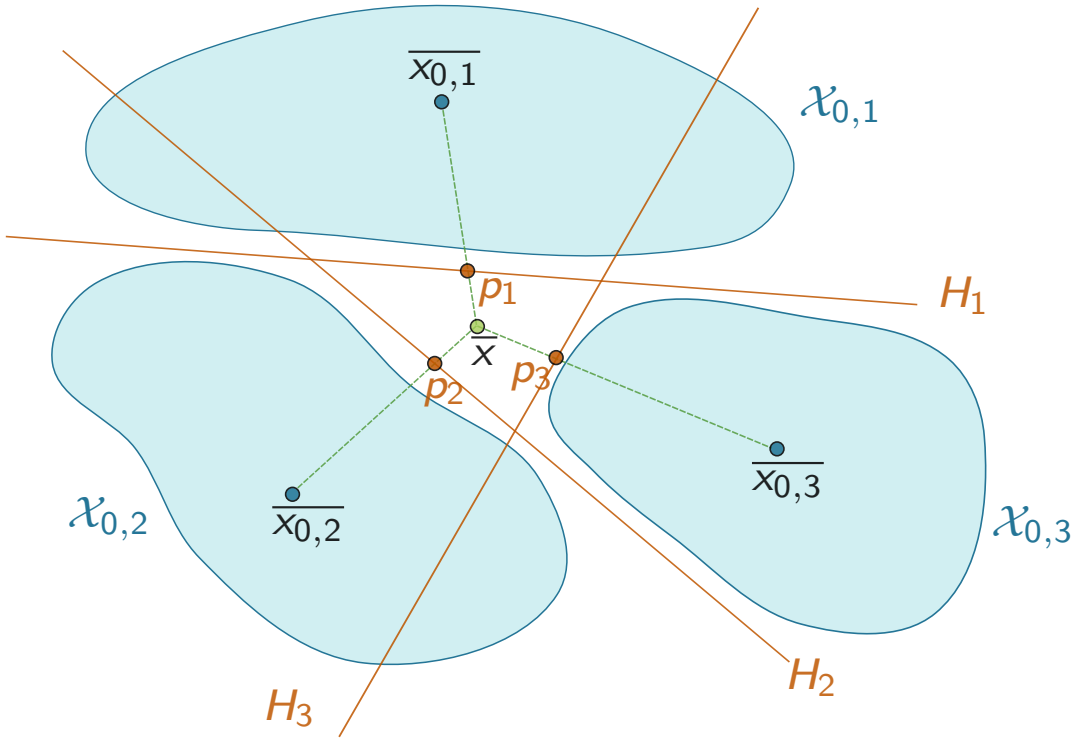
Proof sketch.



□

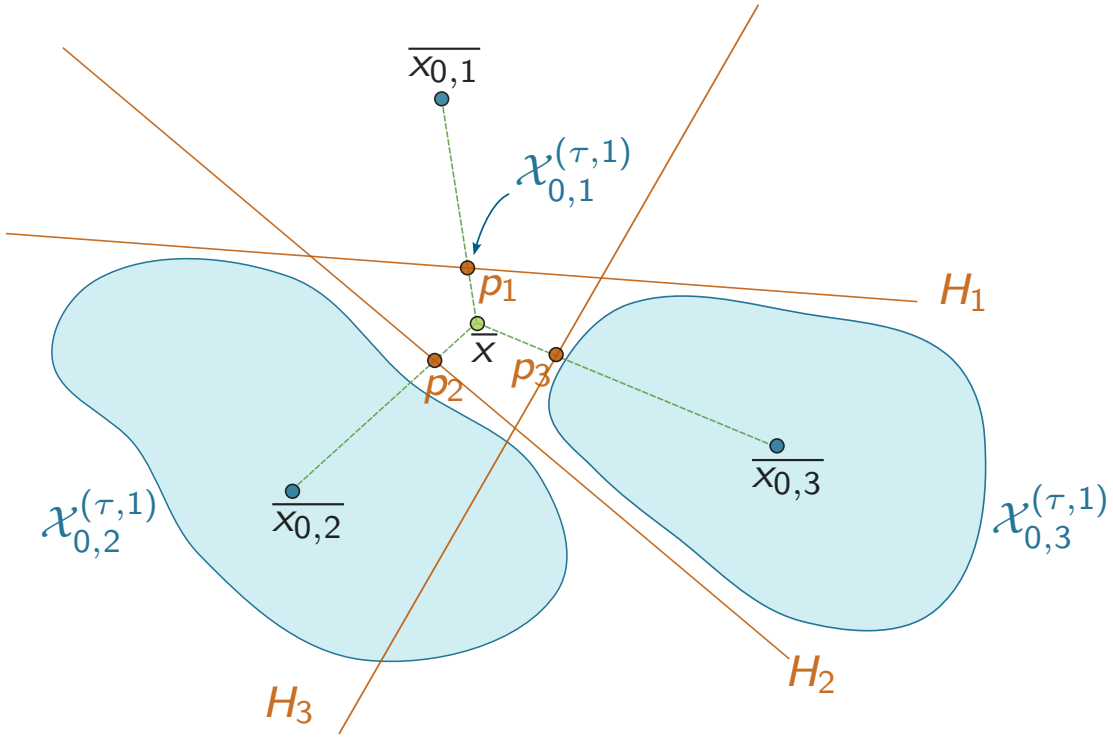
Definition (Chen–E. '25)

We say a data set is **sequentially linearly separable** if there exists an ordering of the classes such that for each $j = 1, \cdots, Q$, there exists a hyperplane H_j and a point $p_j \in H_j$ such that H_j separates $\mathcal{X}_{0,j}$ and $\bigcup_{k<j}\{p_k\} \cup \bigcup_{k>j} \mathcal{X}_{0,k}$.



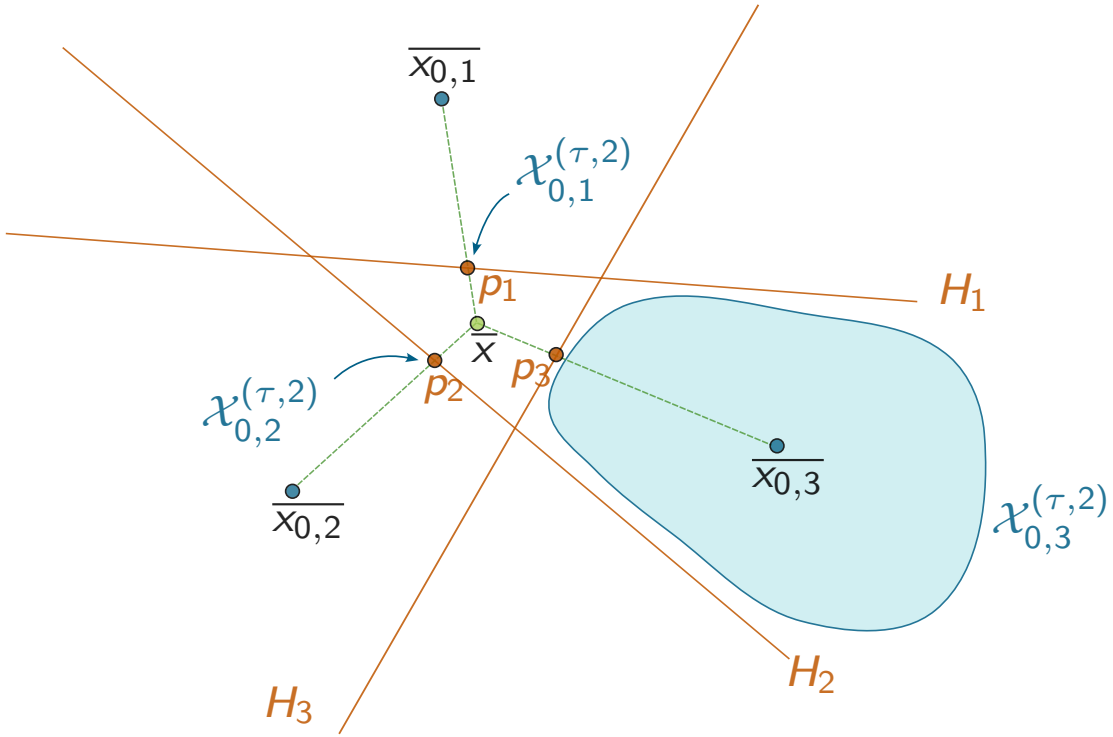
Definition (Chen–E. '25)

We say a data set is **sequentially linearly separable** if there exists an ordering of the classes such that for each $j = 1, \cdots, Q$, there exists a hyperplane H_j and a point $p_j \in H_j$ such that H_j separates $\mathcal{X}_{0,j}$ and $\bigcup_{k<j}\{p_k\} \cup \bigcup_{k>j} \mathcal{X}_{0,k}$.



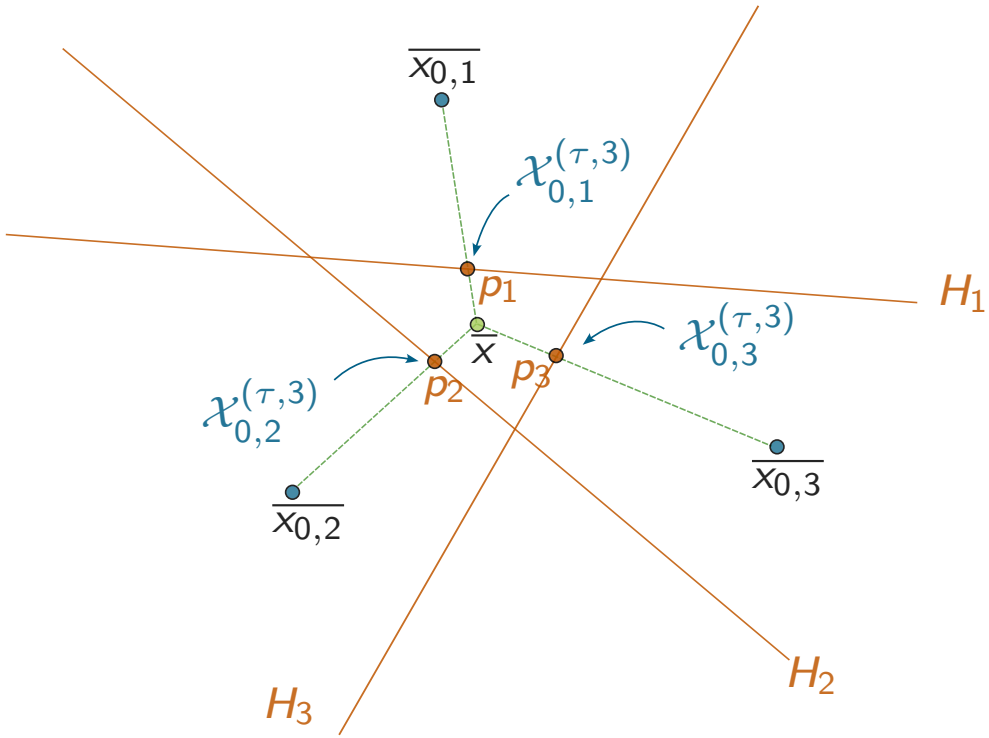
Definition (Chen–E. '25)

We say a data set is **sequentially linearly separable** if there exists an ordering of the classes such that for each $j = 1, \cdots, Q$, there exists a hyperplane H_j and a point $p_j \in H_j$ such that H_j separates $\mathcal{X}_{0,j}$ and $\bigcup_{k<j}\{p_k\} \cup \bigcup_{k>j} \mathcal{X}_{0,k}$.



Definition (Chen–E. '25)

We say a data set is **sequentially linearly separable** if there exists an ordering of the classes such that for each $j = 1, \cdots, Q$, there exists a hyperplane H_j and a point $p_j \in H_j$ such that H_j separates $\mathcal{X}_{0,j}$ and $\bigcup_{k<j}\{p_k\} \cup \bigcup_{k>j} \mathcal{X}_{0,k}$.



We have proved the following:

Theorem (Chen–E. '25)

If the data is sequentially linearly separable, then a ReLU neural network with $Q+1$ layers of size $d_0 = \cdots = d_Q \geq Q$, $d_{Q+1} = Q$, attains a degenerate global minimum with zero training cost, which can be parametrized by

$$\{(\theta_\ell, \nu_\ell, \mu_\ell)\}_{\ell=1}^{Q} \subseteq (0,\pi) \times \mathbb{R}^{d_0} \times (0,1), \tag{4.3}$$

corresponding to triples of angles, normal vectors and a line segment of base points (resp.) describing cones and hyperplanes.

Remarks

- Because we assumed the data to be very structured, the constructions in this paper do not depend on the size of the data set $|\mathcal{X}_0|$, and therefore generalize well.
- Collapsing the classes to their means is one of the defining features of neural collapse.[1]

[1] X. Han, V. Papyan, and D. L. Donoho. *Neural collapse under MSE loss: Proximity to and dynamics on the central path*. ICLR, 2022

Follow-up papers:

- T. Chen, *Derivation of effective gradient flow equations and dynamical truncation of training data in deep learning*, 2025.
- P. M. Ewald, *Explicit neural network classifiers for non-separable data*, 2025.

Thank you!