Critical Points of Deep Linear Networks in $\mathbb{C}^N$

Thesis Defense Presentation

by

Ayush Bharadwaj

Department of Mathematics

San Francisco State University

Critical Points of Deep Linear Networks in $\mathbb{C}^N$

Critical Points of Deep Linear Networks in $\mathbb{C}^N$

Outline

Critical Points of Deep Linear Networks in $\mathbb{C}^N$

Outline

- Neural networks (particularly, *linear networks*)

Outline

- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis

Critical Points of Deep Linear Networks in $\mathbb{C}^N$

Outline

- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis
- Neural network training as solving polynomial systems in algebraic geometry

Critical Points of Deep Linear Networks in $\mathbb{C}^N$

Outline

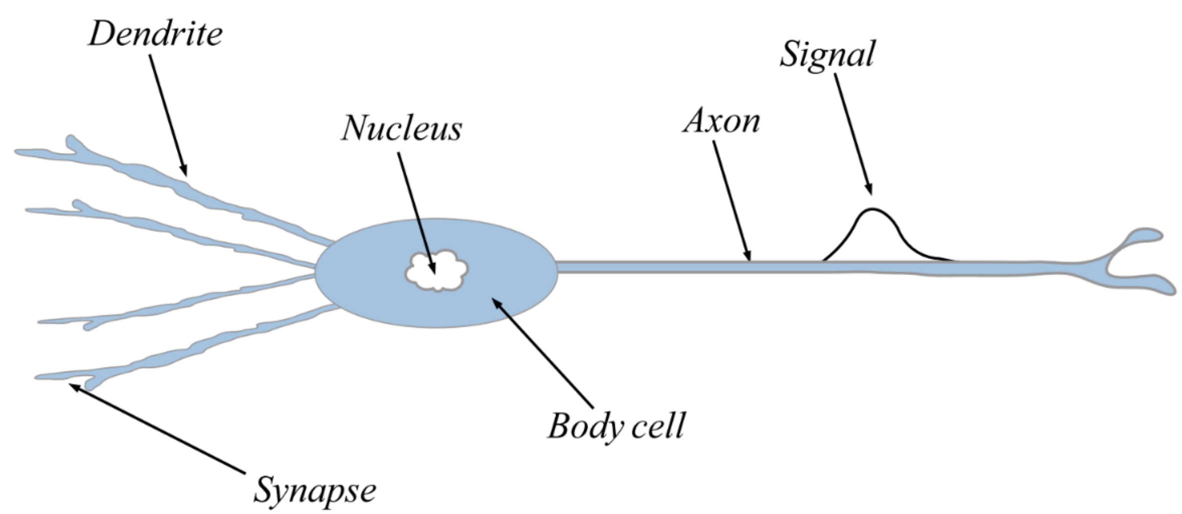
- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis
- Neural network training as solving polynomial systems in algebraic geometry
- Results and contributions

Neural Networks

Neuron

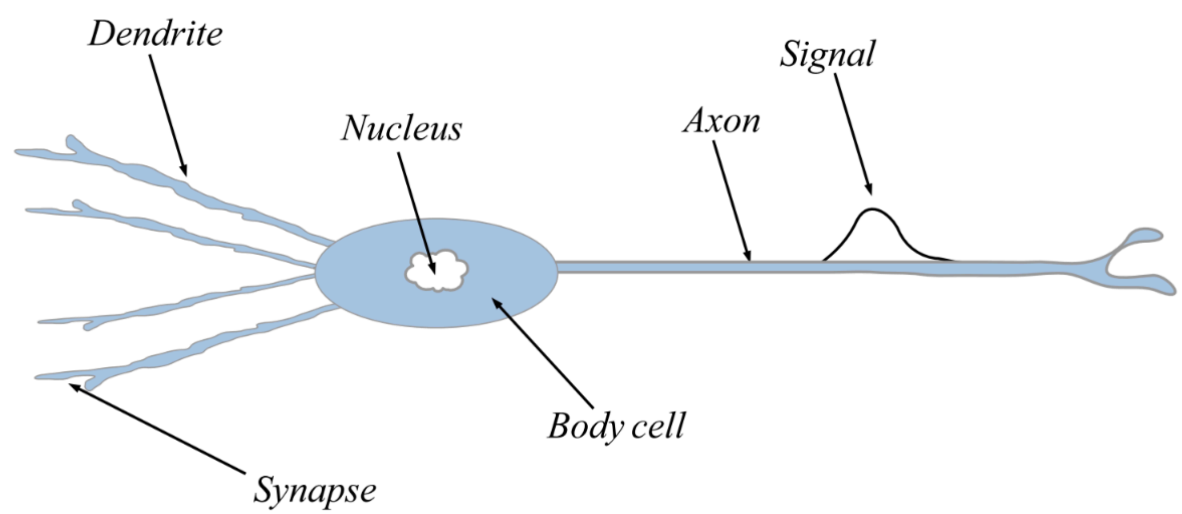
Neural Networks

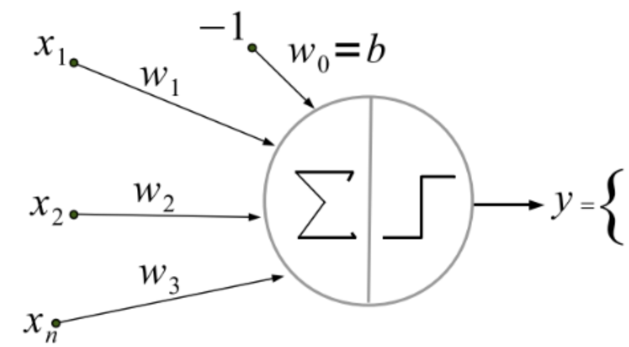
Neuron



Neural Networks

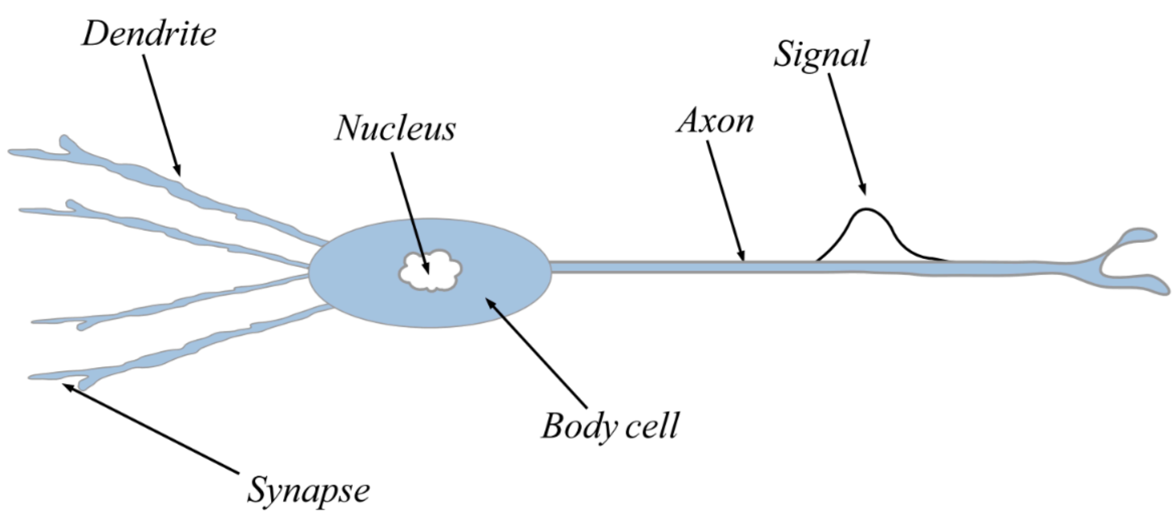
Neuron

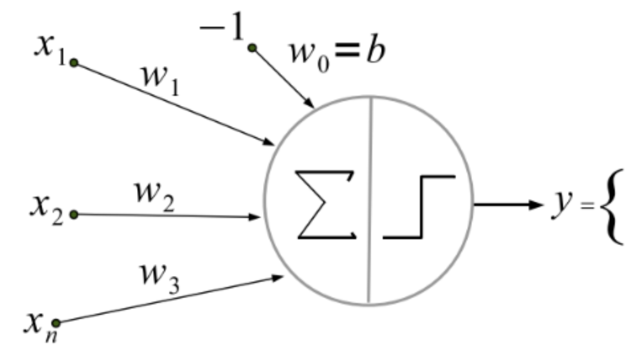




Neural Networks

Neuron

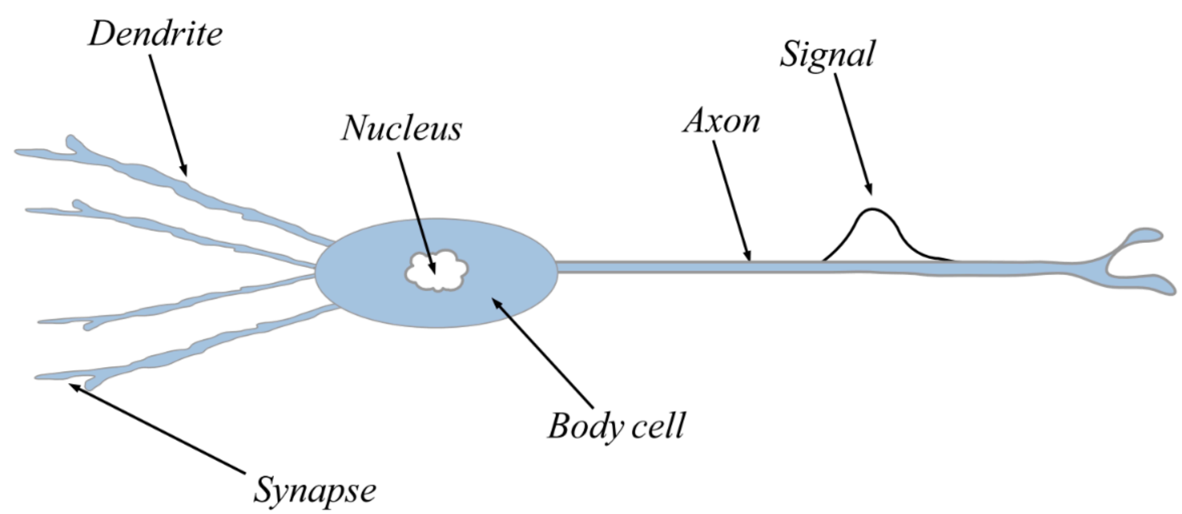


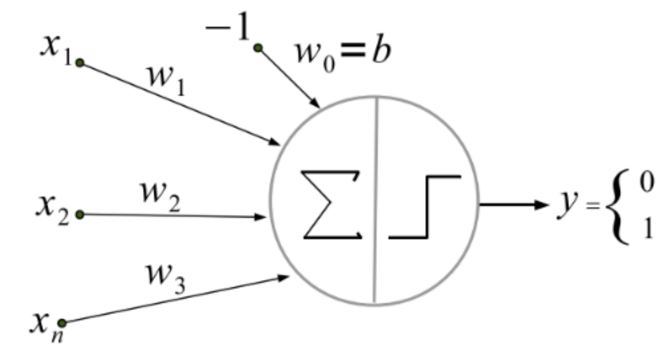


- $f_w(x_1, \ldots, x_n) = \phi\left(\sum_{j=1}^{n} w_j x_j - b\right)$

Neural Networks

Neuron





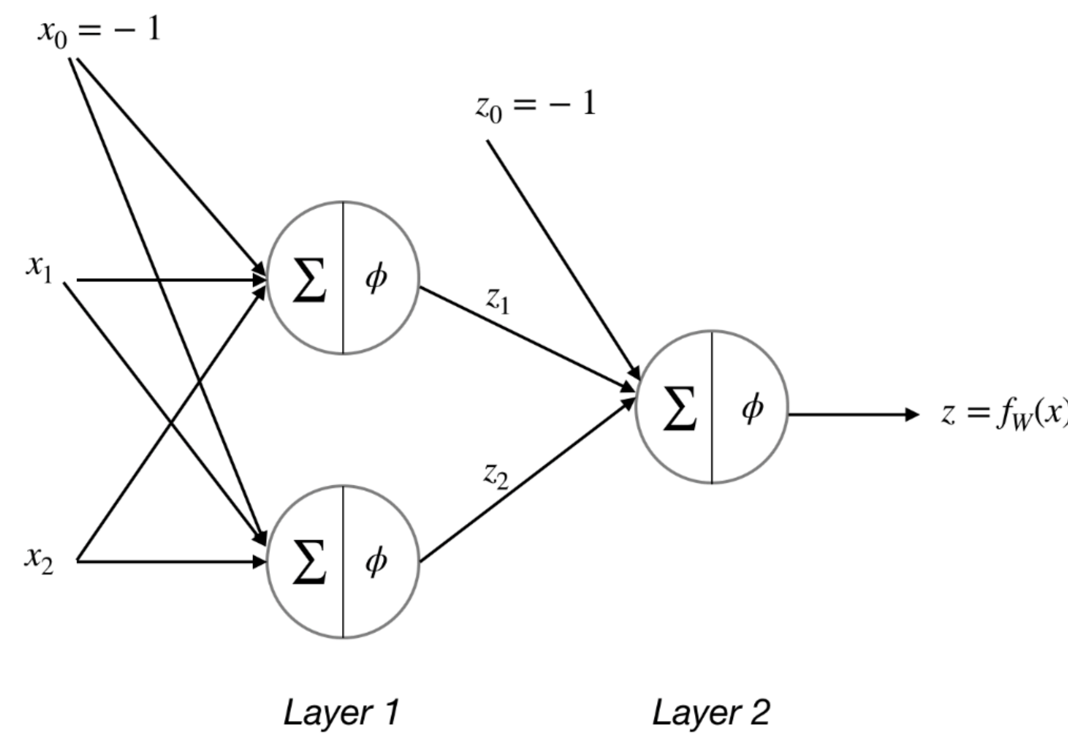
- $f_w(x_1, \ldots, x_n) = \phi\left(\sum_{j=1}^{n} w_j x_j - b\right)$ **(output function)**

Neural Networks

Neural network

Neural Networks

Neural network

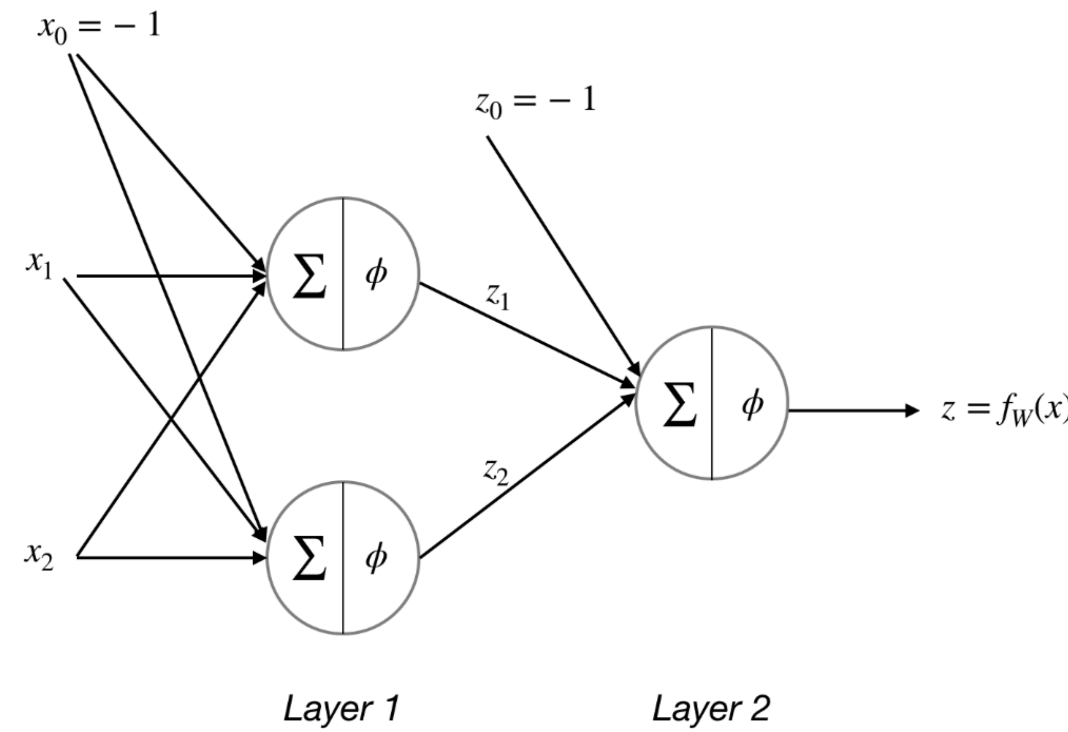


Neural Networks

Neural network

- The weight matrix of the ith layer

$$W_i = \begin{pmatrix} w_{i11} & \cdots & w_{i1n} \\ \vdots & & \vdots \\ w_{ir1} & \cdots & w_{irn} \end{pmatrix}$$

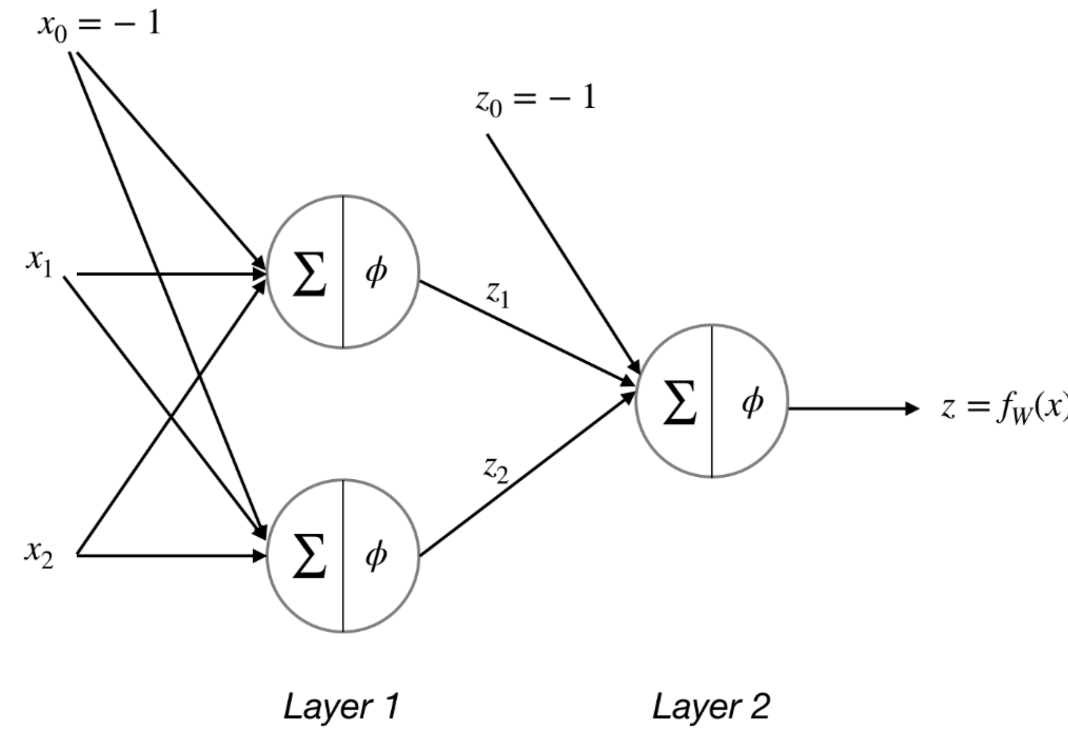


Neural Networks

Neural network

- The weight matrix of the ith layer

$$W_i = \begin{pmatrix} w_{i11} & \cdots & w_{i1n} \\ \vdots & & \vdots \\ w_{ir1} & \cdots & w_{irn} \end{pmatrix}$$

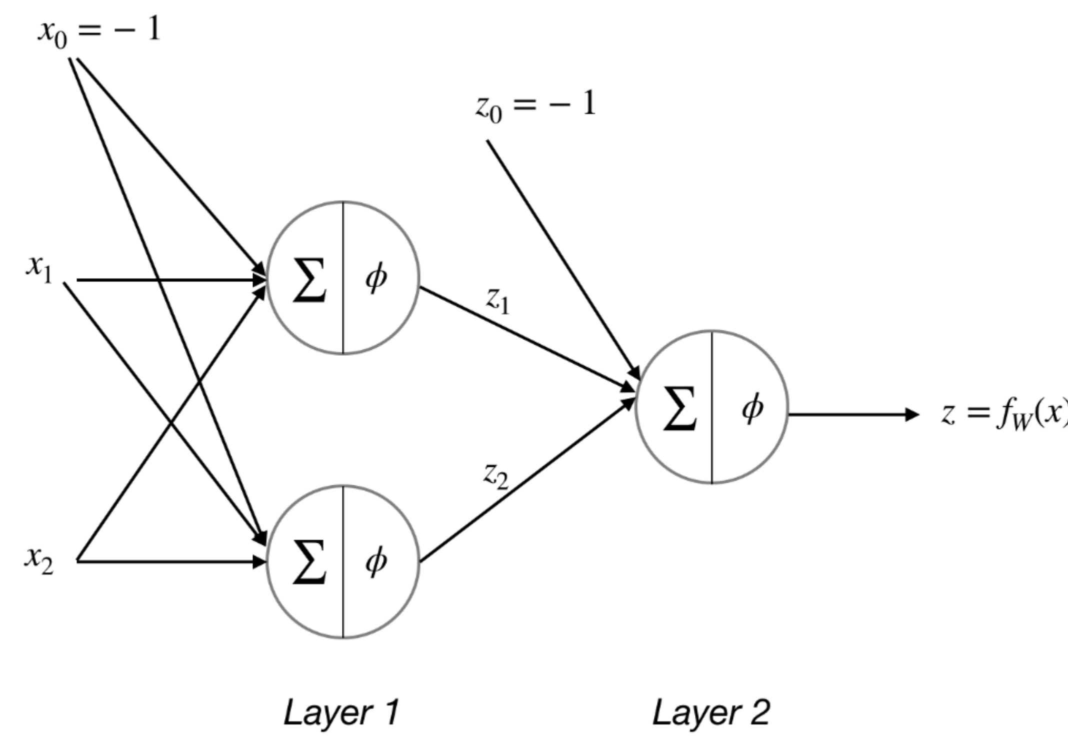


Neural Networks

Neural network

- The weight matrix of the ith layer

$$W_i = \begin{pmatrix} w_{i11} & \cdots & w_{i1n} \\ \vdots & & \vdots \\ w_{ir1} & \cdots & w_{irn} \end{pmatrix}$$



- Output of the ith layer

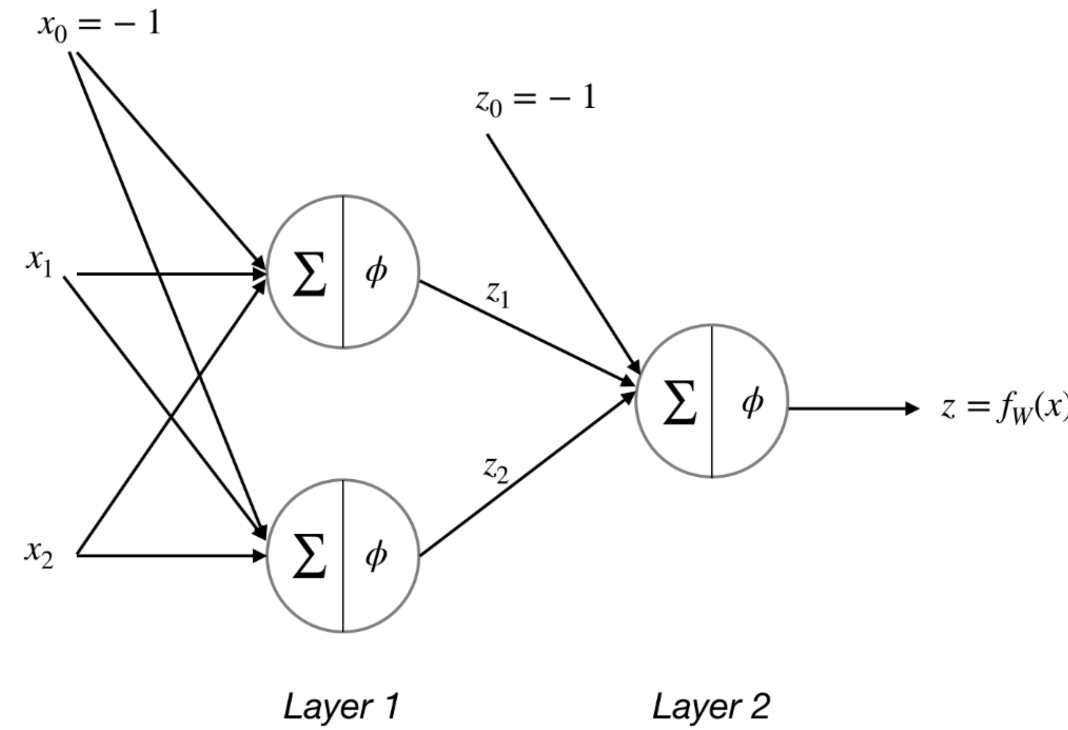
$$z^{(i)} = \phi\left(W_i \begin{bmatrix} -1 \\ z^{(i-1)} \end{bmatrix}\right)$$

Neural Networks

Neural network

- The weight matrix of the ith layer

$$W_i = \begin{pmatrix} w_{i11} & \cdots & w_{i1n} \\ \vdots & & \vdots \\ w_{ir1} & \cdots & w_{irn} \end{pmatrix}$$



- Output of the ith layer

$$z^{(i)} = \phi\Big(W_i \begin{bmatrix} -1 \\ z^{(i-1)} \end{bmatrix}\Big)$$

- Output of the network

$$f_W(x) = z^{(H+1)} = \phi\Big(W_{H+1} \begin{bmatrix} -1 \\ z^{(H)} \end{bmatrix}\Big) \tag{2}$$

Neural Networks

Linear network

Neural Networks

Linear network

We introduce two simplifications in the output function of the neuron:

- Drop the bias term b
- Choose $\phi = \mathbf{I}$

Neural Networks

Linear network

We introduce two simplifications in the output function of the neuron:

- Drop the bias term b
- Choose $\phi = \mathbf{I}$

$$f_w(x_1, ..., x_n) = \phi\left(\sum_{j=1}^{n} w_j x_j - b\right)$$

Neural Networks

Linear network

We introduce two simplifications in the output function of the neuron:

- Drop the bias term b
- Choose $\phi = \mathbf{I}$

$$f_w(x_1, ..., x_n) = \phi\left(\sum_{j=1}^{n} w_j x_j - b\right)$$

This simplifies the network output function:

$$f_W(x) = W_{H+1} W_H \cdots W_2 W_1 x \tag{2}$$

Neural Networks

Linear network

We introduce two simplifications in the output function of the neuron:

- Drop the bias term b
- Choose $\phi = \mathbf{I}$

$$f_w(x_1, ..., x_n) = \phi\left(\sum_{j=1}^{n} w_j x_j - b\right)$$

This simplifies the network output function:

$$\boxed{f_W(x) = W_{H+1} W_H \cdots W_2 W_1 x} \tag{2}$$

Neural Networks

Linear network

We introduce two simplifications in the output function of the neuron:

- Drop the bias term b
- Choose $\phi = \mathbf{I}$

$$f_w(x_1, ..., x_n) = \phi\left(\sum_{j=1}^{n} w_j x_j - b\right)$$

This simplifies the network output function:

$$\boxed{f_W(x) = W_{H+1} W_H \cdots W_2 W_1 x} \tag{2}$$

We want to use f_W to approximate some function $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$

Neural Networks

Training a linear network

Neural Networks

Training a linear network

- We know the values $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$ takes on a finite set of input points $x^{(i)}$, $i = 1, \ldots, m$

Training a linear network

- We know the values $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$ takes on a finite set of input points $x^{(i)}$, $i = 1, ..., m$

- Training examples: $(x^{(i)}, y^{(i)}) := \big(x^{(i)}, y(x^{(i)})\big)$

Neural Networks

Training a linear network

- We know the values $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$ takes on a finite set of input points $x^{(i)}$, $i = 1, ..., m$

- Training examples: $(x^{(i)}, y^{(i)}) := \big(x^{(i)}, y(x^{(i)})\big)$

- Error (or *loss*), $\mathcal{L} : \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by:

$$\mathcal{L}(W) = \frac{1}{2}\sum_{i=1}^{m} ||W_{H+1}W_H...W_2W_1x^{(i)} - y^{(i)}||^2 \tag{3}$$

Neural Networks

Training a linear network

- We know the values $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$ takes on a finite set of input points $x^{(i)}$, $i = 1, ..., m$

- Training examples: $(x^{(i)}, y^{(i)}) := \big(x^{(i)}, y(x^{(i)})\big)$

- Error (or *loss*), $\mathcal{L} : \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by:

$$\mathcal{L}(W) = \frac{1}{2}\sum_{i=1}^{m} ||W_{H+1}W_H...W_2W_1x^{(i)} - y^{(i)}||^2 \tag{3}$$

- We want to solve the minimization problem:

$$\min_{W\in\mathbb{R}^N} \; \mathcal{L}(W) \tag{4}$$

Neural Networks

Training a linear network

- We know the values $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$ takes on a finite set of input points $x^{(i)}$, $i = 1, ..., m$

- Training examples: $(x^{(i)}, y^{(i)}) := \big(x^{(i)}, y(x^{(i)})\big)$

- Error (or *loss*), $\mathcal{L} : \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by:

$$\mathcal{L}(W) = \frac{1}{2} \sum_{i=1}^{m} ||W_{H+1} W_H ... W_2 W_1 x^{(i)} - y^{(i)}||^2 \tag{3}$$

- We want to solve the minimization problem:

$$\min_{W \in \mathbb{R}^N} \; \mathcal{L}(W) \tag{4}$$

Neural Networks

Training a linear network

- We know the values $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$ takes on a finite set of input points $x^{(i)}$, $i = 1, ..., m$

- Training examples: $(x^{(i)}, y^{(i)}) := \big(x^{(i)}, y(x^{(i)})\big)$

Minimization problem in real analysis

- Error (or *loss*), $\mathcal{L} : \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by:

$$\mathcal{L}(W) = \frac{1}{2}\sum_{i=1}^{m} ||W_{H+1}W_H...W_2W_1x^{(i)} - y^{(i)}||^2 \tag{3}$$

- We want to solve the minimization problem:

$$\min_{W\in\mathbb{R}^N} \mathcal{L}(W) \tag{4}$$

Neural Networks

Training a linear network

- We know the values $y : \mathbb{R}^n \longrightarrow \mathbb{R}^p$ takes on a finite set of input points $x^{(i)}$, $i = 1, ..., m$

- Training examples: $(x^{(i)}, y^{(i)}) := \big(x^{(i)}, y(x^{(i)})\big)$

Minimization problem in real analysis

- Error (or *loss*), $\mathcal{L} : \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by:

$$\mathcal{L}(W) = \frac{1}{2}\sum_{i=1}^{m} ||W_{H+1}W_H ... W_2 W_1 x^{(i)} - y^{(i)}||^2 \qquad (3)$$

- We want to solve the minimization problem:

$$\min_{W \in \mathbb{R}^N} \mathcal{L}(W) \qquad (4)$$

Not convex

Neural Networks

Training a linear network

Neural Networks

Training a linear network

Non-convexity of $\mathcal{L}(W)$ presents challenges

Neural Networks

Training a linear network

Non-convexity of $\mathcal{L}(W)$ presents challenges

1. Hard to guarantee that all local minima have been found by the algorithm.

Neural Networks

Training a linear network

Non-convexity of $\mathcal{L}(W)$ presents challenges

1. Hard to guarantee that all local minima have been found by the algorithm.

2. Hard to make assertions about the number and location of minima before hand.

Neural Networks

Training a linear network

Non-convexity of $\mathcal{L}(W)$ presents challenges

1. Hard to guarantee that all local minima have been found by the algorithm.

2. Hard to make assertions about the number and location of minima before hand.

Neural Networks

Training a linear network

Non-convexity of $\mathcal{L}(W)$ presents challenges

1. Hard to guarantee that all local minima have been found by the algorithm.

2. Hard to make assertions about the number and location of minima before hand.

Motivation for an algebraic geometry view

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \ \mathcal{L}(W)$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \ \mathcal{L}(W) \longrightarrow$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \; \mathcal{L}(W) \longrightarrow \nabla\mathcal{L}(W) = 0$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \ \mathcal{L}(W) \longrightarrow \nabla \mathcal{L}(W) = 0$
- $W \in \mathbb{R}^N$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \; \mathcal{L}(W) \longrightarrow \nabla \mathcal{L}(W) = 0$
- $W \in \mathbb{R}^N \longrightarrow$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \ \mathcal{L}(W) \longrightarrow \nabla \mathcal{L}(W) = 0$
- $W \in \mathbb{R}^N \longrightarrow W \in \mathbb{C}^N$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \ \mathcal{L}(W) \longrightarrow \nabla \mathcal{L}(W) = 0$
- $W \in \mathbb{R}^N \longrightarrow W \in \mathbb{C}^N$

We solve:

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \ \mathcal{L}(W) \longrightarrow \nabla\mathcal{L}(W) = 0$
- $W \in \mathbb{R}^N \longrightarrow W \in \mathbb{C}^N$

We solve:

$$\nabla\mathcal{L}(W) = 0, \quad W \in \mathbb{C}^N \tag{5}$$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W\in\mathbb{R}^N} \ \mathcal{L}(W) \longrightarrow \nabla\mathcal{L}(W) = 0$
- $W \in \mathbb{R}^N \longrightarrow W \in \mathbb{C}^N$

We solve:

$$\boxed{\nabla\mathcal{L}(W) = 0, \quad W \in \mathbb{C}^N} \tag{5}$$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

We introduce the following relaxations:

- $\min_{W \in \mathbb{R}^N} \; \mathcal{L}(W) \longrightarrow \nabla \mathcal{L}(W) = 0$
- $W \in \mathbb{R}^N \longrightarrow W \in \mathbb{C}^N$

We solve:

$$\boxed{\nabla \mathcal{L}(W) = 0, \quad W \in \mathbb{C}^N} \tag{5}$$

Polynomial system

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Solutions are precisely the critical points

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

Solutions are precisely the critical points

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2]$$

Solutions are precisely the critical points

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

Solutions are precisely the critical points

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad X = \left(\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right), Y = \left(\begin{smallmatrix} 1 & 3 \\ 2 & 4 \end{smallmatrix}\right).$$

Solutions are
precisely
the critical points

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, Y = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

- $\mathcal{L}(W) = \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = \frac{1}{2} \sum_{i=1}^{2} ||W_2 W_1 x^{(i)} - y^{(i)}||^2$

Solutions are precisely the critical points

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, Y = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

- $\mathcal{L}(W) = \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = \frac{1}{2} \sum_{i=1}^{2} ||W_2 W_1 x^{(i)} - y^{(i)}||^2$
- $\nabla \mathcal{L}(W) = \nabla \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = 0$ gives

Solutions are precisely the critical points

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, Y = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

- $\mathcal{L}(W) = \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = \frac{1}{2} \sum_{i=1}^{2} ||W_2 W_1 x^{(i)} - y^{(i)}||^2$

- $\nabla \mathcal{L}(W) = \nabla \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = 0$ gives

Solutions are precisely the critical points

$$\begin{aligned}
5\alpha_1\beta_1^2 + 5\alpha_1\beta_2^2 + 11\alpha_2\beta_1^2 + 11\alpha_2\beta_2^2 - 7\beta_1 - 10\beta_2 &= 0 \\
11\alpha_1\beta_1^2 + 11\alpha_1\beta_2^2 + 25\alpha_2\beta_1^2 + 25\alpha_2\beta_2^2 - 15\beta_1 - 22\beta_2 &= 0 \\
5\alpha_1^2\beta_1 + 22\alpha_1\alpha_2\beta_1 + 25\alpha_2^2\beta_1 - 7\alpha_1 - 15\alpha_2 &= 0 \\
5\alpha_1^2\beta_2 + 22\alpha_1\alpha_2\beta_2 + 25\alpha_2^2\beta_2 - 10\alpha_1 - 22\alpha_2 &= 0
\end{aligned} \tag{1.13}$$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, Y = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

- $\mathcal{L}(W) = \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = \frac{1}{2} \sum_{i=1}^{2} ||W_2 W_1 x^{(i)} - y^{(i)}||^2$
- $\nabla \mathcal{L}(W) = \nabla \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = 0$ gives

Solutions are precisely the critical points

$$\boxed{\begin{aligned} 5\alpha_1\beta_1^2 + 5\alpha_1\beta_2^2 + 11\alpha_2\beta_1^2 + 11\alpha_2\beta_2^2 - 7\beta_1 - 10\beta_2 = 0 \\ 11\alpha_1\beta_1^2 + 11\alpha_1\beta_2^2 + 25\alpha_2\beta_1^2 + 25\alpha_2\beta_2^2 - 15\beta_1 - 22\beta_2 = 0 \\ 5\alpha_1^2\beta_1 + 22\alpha_1\alpha_2\beta_1 + 25\alpha_2^2\beta_1 - 7\alpha_1 - 15\alpha_2 = 0 \\ 5\alpha_1^2\beta_2 + 22\alpha_1\alpha_2\beta_2 + 25\alpha_2^2\beta_2 - 10\alpha_1 - 22\alpha_2 = 0 \end{aligned}} \tag{1.13}$$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, Y = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

- $\mathcal{L}(W) = \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = \frac{1}{2}\sum_{i=1}^{2} ||W_2 W_1 x^{(i)} - y^{(i)}||^2$
- $\nabla\mathcal{L}(W) = \nabla\mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = 0$ gives

Nice polynomial system

Solutions are precisely the critical points

$$\boxed{\begin{aligned} 5\alpha_1\beta_1^2 + 5\alpha_1\beta_2^2 + 11\alpha_2\beta_1^2 + 11\alpha_2\beta_2^2 - 7\beta_1 - 10\beta_2 &= 0 \\ 11\alpha_1\beta_1^2 + 11\alpha_1\beta_2^2 + 25\alpha_2\beta_1^2 + 25\alpha_2\beta_2^2 - 15\beta_1 - 22\beta_2 &= 0 \\ 5\alpha_1^2\beta_1 + 22\alpha_1\alpha_2\beta_1 + 25\alpha_2^2\beta_1 - 7\alpha_1 - 15\alpha_2 &= 0 \\ 5\alpha_1^2\beta_2 + 22\alpha_1\alpha_2\beta_2 + 25\alpha_2^2\beta_2 - 10\alpha_1 - 22\alpha_2 &= 0 \end{aligned}} \tag{1.13}$$

Neural Networks

Training a linear network

From a problem in Real Analysis to a problem in Algebraic Geometry

Example 1.2 (A 4-weight network). Let us consider a 2-layer network with

$$W_1 = [\alpha_1, \alpha_2] \quad W_2 = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad X = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, Y = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

- $\mathcal{L}(W) = \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = \frac{1}{2} \sum_{i=1}^{2} ||W_2 W_1 x^{(i)} - y^{(i)}||^2$

- $\nabla \mathcal{L}(W) = \nabla \mathcal{L}(\alpha_1, \alpha_2, \beta_1, \beta_2) = 0$ gives

Nice polynomial system

- **Square**
- **Sparse**
- **Deg 2H + 1**

Solutions are precisely the critical points

$$\boxed{\begin{aligned} 5\alpha_1\beta_1^2 + 5\alpha_1\beta_2^2 + 11\alpha_2\beta_1^2 + 11\alpha_2\beta_2^2 - 7\beta_1 - 10\beta_2 &= 0 \\ 11\alpha_1\beta_1^2 + 11\alpha_1\beta_2^2 + 25\alpha_2\beta_1^2 + 25\alpha_2\beta_2^2 - 15\beta_1 - 22\beta_2 &= 0 \\ 5\alpha_1^2\beta_1 + 22\alpha_1\alpha_2\beta_1 + 25\alpha_2^2\beta_1 - 7\alpha_1 - 15\alpha_2 &= 0 \\ 5\alpha_1^2\beta_2 + 22\alpha_1\alpha_2\beta_2 + 25\alpha_2^2\beta_2 - 10\alpha_1 - 22\alpha_2 &= 0 \end{aligned}} \tag{1.13}$$

Algebraic Geometry

Main Ideas

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

2. use these upper bounds to algorithmically find **all** complex solutions

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

2. use these upper bounds to algorithmically find **all** complex solutions

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

2. use these upper bounds to algorithmically find **all** complex solutions

adresses challenge 1

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

2. use these upper bounds to algorithmically find **all** complex solutions

adresses challenge 1

Questions:

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

2. use these upper bounds to algorithmically find **all** complex solutions

adresses challenge 1

Questions:

- How to place upper bounds

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

2. use these upper bounds to algorithmically find **all** complex solutions

adresses challenge 1

Questions:

- How to place upper bounds
- How to solve the system

Algebraic Geometry

Main Ideas

Algebraic geometry provides results and methods to analyze and solve systems of polynomial equations. In particular, it provides us ways to:

1. exploit the monomial structure of a polynomial system to place upper bounds on the number of complex solutions **beforehand**

adresses challenge 2

2. use these upper bounds to algorithmically find **all** complex solutions

adresses challenge 1

Questions:

- How to place upper bounds
- How to solve the system

Homotopy continuation

Algebraic Geometry

Main Ideas

Homotopy Continuation

Algebraic Geometry

Main Ideas

Homotopy Continuation

- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$

Algebraic Geometry

Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$

Algebraic Geometry

Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$

Algebraic Geometry

Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$ **Start system**

Algebraic Geometry

Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$ **Start system**
- $G(x) = 0$ is guaranteed to have at least as many isolated solutions as $F(x) = 0$ and these solutions are known beforehand

Algebraic Geometry

Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$ **Start system**
- $G(x) = 0$ is guaranteed to have at least as many isolated solutions as $F(x) = 0$ and these solutions are known beforehand
- We define a parameterized family of systems:

$$H(x,t) = tG(x) + (1-t)F(x), \quad t \in [0,1]$$

Algebraic Geometry

Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$ **Start system**
- $G(x) = 0$ is guaranteed to have at least as many isolated solutions as $F(x) = 0$ and these solutions are known beforehand
- We define a parameterized family of systems: **Homotopy**

$$H(x, t) = tG(x) + (1 - t)F(x), \quad t \in [0, 1]$$

Algebraic Geometry

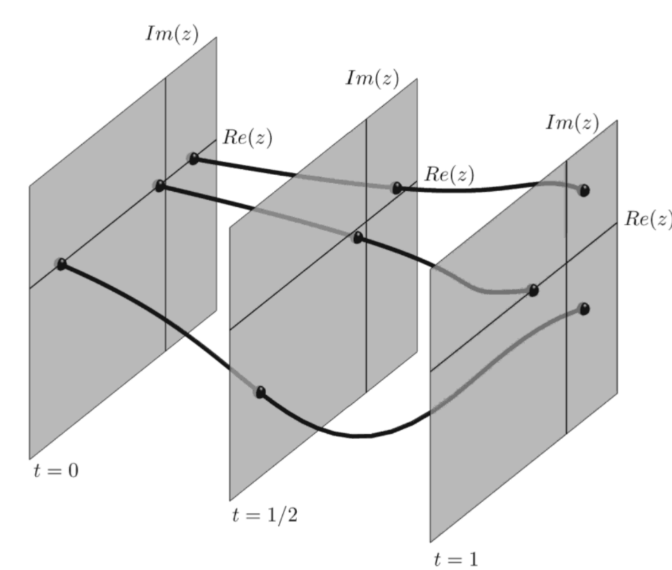
Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$ **Start system**
- $G(x) = 0$ is guaranteed to have at least as many isolated solutions as $F(x) = 0$ and these solutions are known beforehand
- We define a parameterized family of systems:

$$H(x,t) = tG(x) + (1-t)F(x), \quad t \in [0,1]$$

Homotopy



Algebraic Geometry

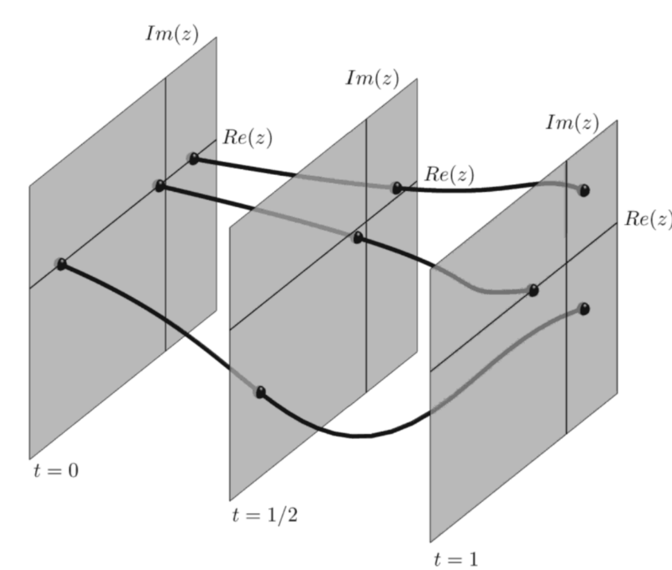
Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$ **Start system**
- $G(x) = 0$ is guaranteed to have at least as many isolated solutions as $F(x) = 0$ and these solutions are known beforehand
- We define a parameterized family of systems:

$$H(x,t) = tG(x) + (1-t)F(x), \quad t \in [0,1]$$

Homotopy



A tighter upper bound means fewer paths to track and therefore provides a more efficient way to solve the target system

Algebraic Geometry

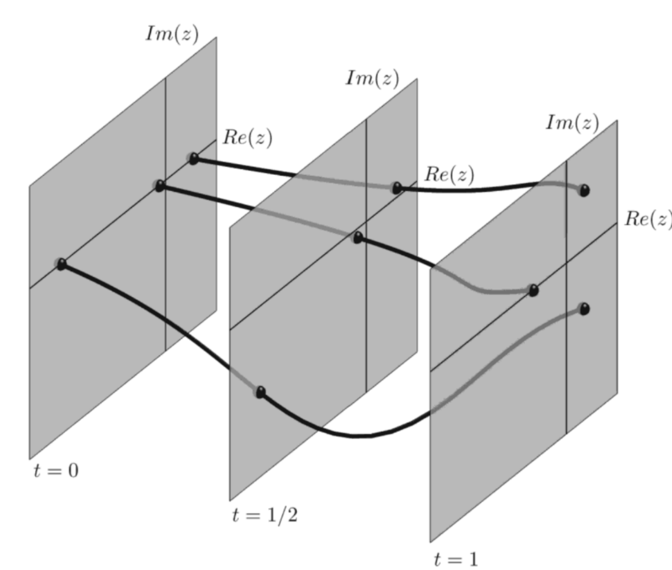
Main Ideas

Homotopy Continuation

- Suppose we want to solve the system $F(x) = [f_1(x), ..., f_N(x)] = 0$ **Target system**
- We generate another polynomial system $G(x) = [g_1(x), ..., g_N(x)] = 0$ **Start system**
- $G(x) = 0$ is guaranteed to have at least as many isolated solutions as $F(x) = 0$ and these solutions are known beforehand
- We define a parameterized family of systems:

Homotopy

$$H(x,t) = tG(x) + (1-t)F(x), \quad t \in [0,1]$$



A tighter upper bound means fewer paths to track and therefore provides a more efficient way to solve the target system

Algebraic Geometry

Main Ideas

Well known upper bounds

Algebraic Geometry

Main Ideas

Well known upper bounds

Theorem 2.13 (Classical Bezout Bound). *Let $f_1, ..., f_N$ be polynomials in $\mathbb{C}[x_1, ..., x_N]$. Then the number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ is bounded above by the product $deg(f_1) \cdots deg(f_N)$.*

Algebraic Geometry

Main Ideas

Well known upper bounds

Theorem 2.13 (Classical Bezout Bound). *Let $f_1, ..., f_N$ be polynomials in $\mathbb{C}[x_1, ..., x_N]$. Then the number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ is bounded above by the product $deg(f_1) \cdots deg(f_N)$.*

Theorem 2.22 (Bernstein's Theorem). *Let $f_1, ..., f_N \in \mathbb{C}[x_1, ..., x_N]$ be Laurent polynomials with Newton polytopes $Q_1, ..., Q_N$. The number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ in $(\mathbb{C}^*)^N$ is bounded above by the mixed volume $\mathcal{M}(Q_1, ..., Q_N)$.*

Algebraic Geometry

Main Ideas

Well known upper bounds

Theorem 2.13 (**Classical Bezout Bound**). *Let $f_1, ..., f_N$ be polynomials in $\mathbb{C}[x_1, ..., x_N]$. Then the number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ is bounded above by the product $deg(f_1) \cdots deg(f_N)$.*

Theorem 2.22 (**Bernstein's Theorem**). *Let $f_1, ..., f_N \in \mathbb{C}[x_1, ..., x_N]$ be Laurent polynomials with Newton polytopes $Q_1, ..., Q_N$. The number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ in $(\mathbb{C}^*)^N$ is bounded above by the mixed volume $\mathcal{M}(Q_1, ..., Q_N)$.*

Question: Can we do better?

Algebraic Geometry

Main Ideas

Well known upper bounds

Theorem 2.13 (Classical Bezout Bound). *Let $f_1, ..., f_N$ be polynomials in $\mathbb{C}[x_1, ..., x_N]$. Then the number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ is bounded above by the product $deg(f_1) \cdots deg(f_N)$.*

Theorem 2.22 (Bernstein's Theorem). *Let $f_1, ..., f_N \in \mathbb{C}[x_1, ..., x_N]$ be Laurent polynomials with Newton polytopes $Q_1, ..., Q_N$. The number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ in $(\mathbb{C}^*)^N$ is bounded above by the mixed volume $\mathcal{M}(Q_1, ..., Q_N)$.*

Question: Can we do better? Yes!

Algebraic Geometry

Main Ideas

Well known upper bounds

Theorem 2.13 (Classical Bezout Bound). *Let $f_1, ..., f_N$ be polynomials in $\mathbb{C}[x_1, ..., x_N]$. Then the number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ is bounded above by the product $deg(f_1) \cdots deg(f_N)$.*

Theorem 2.22 (Bernstein's Theorem). *Let $f_1, ..., f_N \in \mathbb{C}[x_1, ..., x_N]$ be Laurent polynomials with Newton polytopes $Q_1, ..., Q_N$. The number of isolated solutions of the system $f_1(x) = ... = f_N(x) = 0$ in $(\mathbb{C}^*)^N$ is bounded above by the mixed volume $\mathcal{M}(Q_1, ..., Q_N)$.*

Question: Can we do better? Yes!

Results

Number of complex critical points

Critical points with all non-zero weights

Results

Number of complex critical points

Critical points with all non-zero weights

Proposition 3.6 (upper bound on solutions in $(\mathbb{C}^*)^N$, $\mathcal{B}_{\mathbb{C}^*}$). *Consider a linear network with* $H = 1$, $m = 1$, $d_x = n$, $d_y = p$ *and* $d_1 = d$. *Let* $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix} \right)$ *denote a solution to the gradient polynomial system (3.1). Then, there are at most*

$$\mathcal{B}_{\mathbb{C}^*} = (4p)^d$$

solutions for which $a_{1,1}, \ldots, a_{d,n} \in \mathbb{C}^*$ *and* $b_{1,1}, \ldots, b_{p,d} \in \mathbb{C}^*$.

Results

Number of complex critical points

Critical points with all non-zero weights

Proposition 3.6 (upper bound on solutions in $(\mathbb{C}^*)^N$, $\mathcal{B}_{\mathbb{C}^*}$). *Consider a linear network with $H = 1$, $m = 1$, $d_x = n$, $d_y = p$ and $d_1 = d$. Let* $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix} \right)$ *denote a solution to the gradient polynomial system (3.1). Then, there are at most*

$$\mathcal{B}_{\mathbb{C}^*} = (4p)^d$$

solutions for which $a_{1,1}, ..., a_{d,n} \in \mathbb{C}^*$ and $b_{1,1}, ..., b_{p,d} \in \mathbb{C}^*$.

Results

Number of complex critical points

All complex critical points

Results

Number of complex critical points

All complex critical points

Theorem 3.9 (upper bound on complex critical points, $\mathcal{B}_{\mathbb{C}}$)**.** *Consider a linear network with $H = 1$, $m = 1$, $d_x = n$, $d_y = p$ and $d_1 = d$. This network has at most*

$$\mathcal{B}_{\mathbb{C}} = (1 + 4p)^d$$

complex critical points.

Results

Number of complex critical points

All complex critical points

Theorem 3.9 (upper bound on complex critical points, $\mathcal{B}_{\mathbb{C}}$). *Consider a linear network with $H = 1$, $m = 1$, $d_x = n$, $d_y = p$ and $d_1 = d$. This network has at most*

$$\mathcal{B}_{\mathbb{C}} = (1 + 4p)^d$$

complex critical points.

No	d	d_x	d_y	N	CBB	BKK	$\mathcal{B}_{\mathbb{C}}$	$\mathcal{B}_{\mathbb{C}^*}$	$N_{\mathbb{C}}$	$N_{\mathbb{C}^*}$	$max\{N_{\mathbb{R}}\}$
1	1	1	1	2	9	5	5	4	5	4	3
2	1	2	1	3	27	9	5	4	5	4	3
3	1	3	1	4	81	13	5	4	5	4	3
4	1	1	2	3	27	9	9	8	9	8	3
5	1	2	2	4	81	33	9	8	9	8	3
6	1	3	2	5	243	73	9	8	9	8	3
7	1	1	3	4	81	13	13	12	13	12	3
8	1	2	3	5	243	73	13	12	13	12	3
9	1	3	3	6	729	245	13	12	13	12	3
10	2	1	1	4	81	25	25	16	9	0	4
11	2	2	1	6	729	81	25	16	9	0	5
12	2	3	1	8	6561	169	25	16	9	0	5
13	2	1	2	6	729	81	81	64	33	16	9
14	2	2	2	8	6561	1089	81	64	33	16	9
15	2	3	2	10	59049	5329	81	64	33	16	9
16	2	1	3	8	6561	169	169	144	73	48	9
17	2	2	3	10	59049	5329	169	144	73	48	9
18	2	3	3	12	531441	60025	169	144	73	48	9
19	3	1	1	6	729	125	125	64	13	0	7
20	3	2	1	9	19683	729	125	64	13	0	7
21	3	3	1	12	531441	2197	125	64	13	0	7
22	3	1	2	9	19683	729	729	512	73	0	19
23	3	2	2	12	531441	35937	729	512	73	0	19
24	3	3	2	15	14348907	389017	729	512	73	0	19
25	3	1	3	12	531441	2197	2197	1728	245	64	27
26	3	2	3	15	14348907	389017	2197	1728	245	64	27

Table 3.1: Case: $H = 1, m = 1$. Comparison of upper bounds on the number of complex critical points of a linear network. $d =$ number of neurons in each layer, $d_x =$ input dimension and $d_y =$ output dimension. $N =$ total number of weights in the network. CBB and BKK refer to the classical Bezout bound and the BKK bound respectively. $\mathcal{B}_{\mathbb{C}}$ and $\mathcal{B}_{\mathbb{C}^*}$ refer to the new bounds on the number of critical points in $(\mathbb{C})^N$ and $(\mathbb{C}^*)^N$ respectively. $N_{\mathbb{C}}$ and $N_{\mathbb{C}^*}$ refer to the actual number of critical points in $(\mathbb{C})^N$ and $(\mathbb{C}^*)^N$ respectively. $max\{N_{\mathbb{R}}\}$ = maximum number of real solutions observed within each sample.

Results

Location of complex critical points

Critical points with some zero weights lie on particular coordinate subspaces

Proposition 3.1 (no stray zeros in W_1). *Consider a linear network with $H = 1, m = 1, d_x = n, d_y = p$ and $d_1 = d$. Let $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix}\right)$ denote a solution to the regularized gradient polynomial system (3.1). If $a_{i,j} = 0$, then $a_{i,s} = 0$ for all $s = 1, \ldots, n$.*

Proposition 3.3 (no stray zeros in W_2). *Consider a linear network with $H = 1, m = 1, d_x = n, d_y = p$ and $d_1 = d$. Let $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix}\right)$ denote a solution to the regularized gradient polynomial system (3.1). Then,*

$$b_{k,i} = 0 \implies b_{\cdot,i} = 0$$

Results

Location of complex critical points

Critical points with some zero weights lie on particular coordinate subspaces

Proposition 3.1 (no stray zeros in W_1). *Consider a linear network with $H=1, m=1, d_x=n, d_y=p$ and $d_1=d$. Let $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix}\right)$ denote a solution to the regularized gradient polynomial system (3.1). If $a_{i,j}=0$, then $a_{i,s}=0$ for all $s=1,\dots,n$.*

Proposition 3.3 (no stray zeros in W_2). *Consider a linear network with $H=1, m=1, d_x=n, d_y=p$ and $d_1=d$. Let $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix}\right)$ denote a solution to the regularized gradient polynomial system (3.1). Then,*

$$b_{k,i} = 0 \implies b_{\cdot,i} = 0$$

Results

Location of complex critical points

Critical points with some zero weights lie on particular coordinate subspaces

Proposition 3.2 (null rows of W_1 match null columns of W_2). *Consider a linear network with* $H = 1$, $m = 1$, $d_x = n$, $d_y = p$ *and* $d_1 = d$. *Let* $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix} \right)$ *denote a solution to the regularized gradient polynomial system (3.1). Then,*

$$a_{i,\cdot} = 0 \iff b_{\cdot,i} = 0$$

Results

Location of complex critical points

Critical points with some zero weights lie on particular coordinate subspaces

Proposition 3.2 (null rows of W_1 match null columns of W_2)**.** *Consider a linear network with* $H = 1$, $m = 1$, $d_x = n$, $d_y = p$ *and* $d_1 = d$. *Let* $(W_1, W_2) = \left(\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ & \vdots & \\ a_{d,1} & \cdots & a_{d,n} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,d} \\ & \vdots & \\ b_{p,1} & \cdots & b_{p,d} \end{pmatrix}\right)$ *denote a solution to the regularized gradient polynomial system (3.1). Then,*

$$a_{i,\cdot} = 0 \iff b_{\cdot,i} = 0$$

Results

Location of complex critical points

Critical points with some zero weights lie on particular coordinate subspaces

```
###################### di=2, H=1, m=1, dx=2, dy=2 ####################

W2 W1
0 1  0 0
0 1  1 1
subspace dim: 4
solution count: 8

W2 W1
1 1  1 1
1 1  1 1
subspace dim: 8
solution count: 16

W2 W1
1 0  1 1
1 0  0 0
subspace dim: 4
solution count: 8

W2 W1
0 0  0 0
0 0  0 0
subspace dim: 0
solution count: 1
```

Further research

3. run experiments to check if increasing m causes BBK bound to be attained? for what value of m for a given architecture?

4. Conversely, can we show that the BKK bound is never reached? Even though our systems are non-generic, we might be able to solve the corresponding facial system may have no solutions, in which case, the BKK bound is a strict bound.

5. prove zero patterns for $H > 1$

6. extend table for H = 1, m=1 but higher values if d_x, d_y, d_i by solving reduced systems.

Recap

$\mathscr{B}_{\mathbb{C}}$ $\mathscr{B}_{\mathbb{C}^*}$

Recap

- Neural networks (particularly, *linear networks*)

$\mathscr{B}_{\mathbb{C}}$ $\mathscr{B}_{\mathbb{C}^*}$

Recap

- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis

$\mathscr{B}_{\mathbb{C}}$ $\mathscr{B}_{\mathbb{C}^*}$

Recap

- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis
- Neural network training as solving polynomial systems in algebraic geometry

$\mathscr{B}_{\mathbb{C}}$ $\mathscr{B}_{\mathbb{C}^*}$

Recap

- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis
- Neural network training as solving polynomial systems in algebraic geometry
- Results:

$\mathscr{B}_{\mathbb{C}}$ $\mathscr{B}_{\mathbb{C}^*}$

Recap

- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis
- Neural network training as solving polynomial systems in algebraic geometry
- Results:
  - New upper bounds $\mathscr{B}_{\mathbb{C}}, \mathscr{B}_{\mathbb{C}^*}$ on the number of complex critical points of 1-hidden layer networks

Recap

- Neural networks (particularly, *linear networks*)
- Neural network training as a minimization problem in real analysis
- Neural network training as solving polynomial systems in algebraic geometry
- Results:
  - New upper bounds $\mathscr{B}_{\mathbb{C}}, \mathscr{B}_{\mathbb{C}^*}$ on the number of complex critical points of 1-hidden layer networks
  - Structure in the location of complex critical points with some zero weights