Convex Analysis of Non-Convex Neural Networks

Aaron Mishkin

Supervised by Mert Pilanci

Stanford University

1. Motivation: Convexity and Deep Learning

Motivation: Thirteen Years Since AlexNet

13 Years Ago: AlexNet won ILSVRC 2012 and started the modern “deep learning” era of machine learning.

https://image-net.org/challenges/LSVRC/2012/results.html#abstract

Motivation: Thirteen Years Since AlexNet

13 Years Ago: AlexNet won ILSVRC 2012 and started the modern "deep learning" era of machine learning.

AlexNet improved over the next best model by $\approx 10\%$ (top-5).

https://image-net.org/challenges/LSVRC/2012/results.html#abstract

Motivation: Thirteen Years Since AlexNet

13 Years Ago: AlexNet won ILSVRC 2012 and started the modern “deep learning” era of machine learning.

AlexNet improved over the next best model by $\approx 10\%$ (top-5).

Key Techniques:

https://image-net.org/challenges/LSVRC/2012/results.html#abstract

Motivation: Thirteen Years Since AlexNet

13 Years Ago: AlexNet won ILSVRC 2012 and started the modern "deep learning" era of machine learning.

AlexNet improved over the next best model by $\approx 10\%$ (top-5).

Key Techniques:

- "a large, deep convolutional neural network"

https://image-net.org/challenges/LSVRC/2012/results.html#abstract

Motivation: Thirteen Years Since AlexNet

13 Years Ago: AlexNet won ILSVRC 2012 and started the modern "deep learning" era of machine learning.

AlexNet improved over the next best model by $\approx 10\%$ (top-5).

Key Techniques:

- "a large, deep convolutional neural network"
- "a very efficient GPU implementation of convolutional nets"

https://image-net.org/challenges/LSVRC/2012/results.html#abstract

Motivation: Thirteen Years Since AlexNet

13 Years Ago: AlexNet won ILSVRC 2012 and started the modern "deep learning" era of machine learning.

AlexNet improved over the next best model by $\approx 10\%$ (top-5).

Key Techniques:

- "a large, deep convolutional neural network"
- "a very efficient GPU implementation of convolutional nets"
- "'dropout', a recently-developed regularization method that proved to be very effective"

https://image-net.org/challenges/LSVRC/2012/results.html#abstract

Motivation: Thirteen Years Since AlexNet

13 Years Ago: AlexNet won ILSVRC 2012 and started the modern "deep learning" era of machine learning.

AlexNet improved over the next best model by $\approx 10\%$ (top-5).

Key Techniques:

- "a large, deep convolutional neural network"
- "a very efficient GPU implementation of convolutional nets"
- "'dropout', a recently-developed regularization method that proved to be very effective"

Not so different from today...

https://image-net.org/challenges/LSVRC/2012/results.html#abstract

Motivation: ImageNet Today

AlexNet won with 84.69% top-five accuracy [KSH12].

Motivation: ImageNet Today

AlexNet won with 84.69% top-five accuracy [KSH12].

Today, models get 99.02% top-5 accuracy [Yua+21]!

(Using all sorts of tricks like pre-training, transformers, etc.)

Motivation: DALL·E 2

We've developed amazing deep learning tools since AlexNet.

https://openai.com/dall-e-2/

Motivation: DALL·E 2

We've developed amazing deep learning tools since AlexNet.

Generated by DALL·E 2

A bowl of soup that is a portal to another dimension as digital art.

https://openai.com/dall-e-2/

Motivation: Cost of Training DALL·E 2

DALL·E 2 has 5.5 billion parameters and took billions of Adam iterations to fit [Ram+22].

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

Motivation: Cost of Training DALL·E 2

DALL·E 2 has 5.5 billion parameters and took billions of Adam iterations to fit [Ram+22].

But this is small compared to recent LLMs!

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

Motivation: Cost of Training DALL·E 2

DALL·E 2 has 5.5 billion parameters and took billions of Adam iterations to fit [Ram+22].

But this is small compared to recent LLMs!

Consider OpenAI's GPT-4 model:

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

Motivation: Cost of Training DALL·E 2

DALL·E 2 has 5.5 billion parameters and took billions of Adam iterations to fit [Ram+22].

But this is small compared to recent LLMs!

Consider OpenAI's GPT-4 model:

- GPT-4 has 1.8 trillion parameters;

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

Motivation: Cost of Training DALL·E 2

DALL·E 2 has 5.5 billion parameters and took billions of Adam iterations to fit [Ram+22].

But this is small compared to recent LLMs!

Consider OpenAI's GPT-4 model:

- GPT-4 has 1.8 trillion parameters;
- It was trained on ≈ 13 trillion tokens;

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

Motivation: Cost of Training DALL·E 2

DALL·E 2 has 5.5 billion parameters and took billions of Adam iterations to fit [Ram+22].

But this is small compared to recent LLMs!

Consider OpenAI's GPT-4 model:

- GPT-4 has 1.8 trillion parameters;
- It was trained on ≈ 13 trillion tokens;
- Training used 25,000 A100s for 90 to 100 days;

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

Motivation: Cost of Training DALL·E 2

DALL·E 2 has 5.5 billion parameters and took billions of Adam iterations to fit [Ram+22].

But this is small compared to recent LLMs!

Consider OpenAI's GPT-4 model:

- GPT-4 has 1.8 trillion parameters;
- It was trained on ≈ 13 trillion tokens;
- Training used 25,000 A100s for 90 to 100 days;
- At \$1 per A100 hour, GPT-4 cost $\approx \$63$ million dollars.

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

Motivation: Challenges of Non-Convexity

Takeaway: Modern deep learning models are huge and extremely expensive to train, but they have tremendous impact.

www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

Motivation: Challenges of Non-Convexity

Takeaway: Modern deep learning models are huge and extremely expensive to train, but they have tremendous impact.

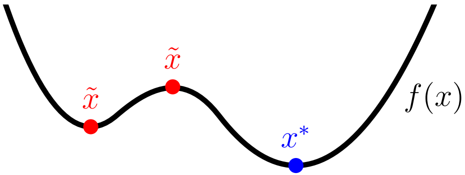
$\hookrightarrow$ ChatGPT is the fastest-adopted piece of software in history!

www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

Motivation: Challenges of Non-Convexity

Takeaway: Modern deep learning models are huge and extremely expensive to train, but they have tremendous impact.

$\hookrightarrow$ ChatGPT is the fastest-adopted piece of software in history!



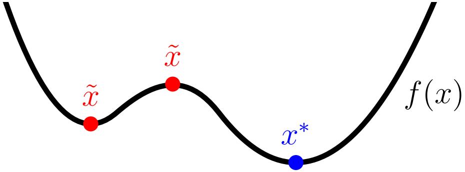
Challenges of Non-Convexity:

www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

Motivation: Challenges of Non-Convexity

Takeaway: Modern deep learning models are huge and extremely expensive to train, but they have tremendous impact.

$\hookrightarrow$ ChatGPT is the fastest-adopted piece of software in history!



Challenges of Non-Convexity:

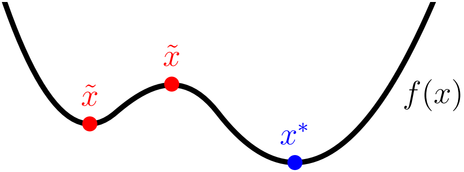
- Optimization: saddle-points, local minima, slow convergence.

www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

Motivation: Challenges of Non-Convexity

Takeaway: Modern deep learning models are huge and extremely expensive to train, but they have tremendous impact.

$\hookrightarrow$ ChatGPT is the fastest-adopted piece of software in history!



Challenges of Non-Convexity:

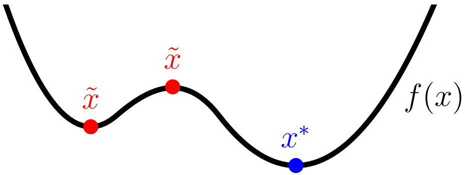
- Optimization: saddle-points, local minima, slow convergence.
- Optimality Conditions: stationarity $\not\Longrightarrow$ optimality.

www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

Motivation: Challenges of Non-Convexity

Takeaway: Modern deep learning models are huge and extremely expensive to train, but they have tremendous impact.

$\hookrightarrow$ ChatGPT is the fastest-adopted piece of software in history!



Challenges of Non-Convexity:

- Optimization: saddle-points, local minima, slow convergence.
- Optimality Conditions: stationarity $\not\Rightarrow$ optimality.
- Mathematical Tools: No subgradients, no separating hyperplanes, (usually) non-zero duality gap.

www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app

Motivation: Convexity and Deep Learning

Key Question: How can we overcome non-convexity to get better optimization and fundamental theory for neural networks?

Motivation: Convexity and Deep Learning

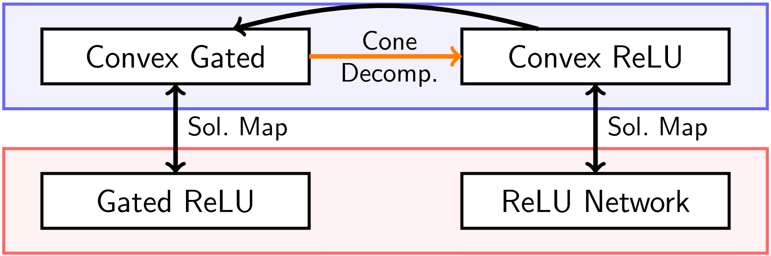
Key Question: How can we overcome non-convexity to get better optimization and fundamental theory for neural networks?

This Thesis: By creating and studying convex reformulations.

Motivation: Convexity and Deep Learning

Key Question: How can we overcome non-convexity to get better optimization and fundamental theory for neural networks?

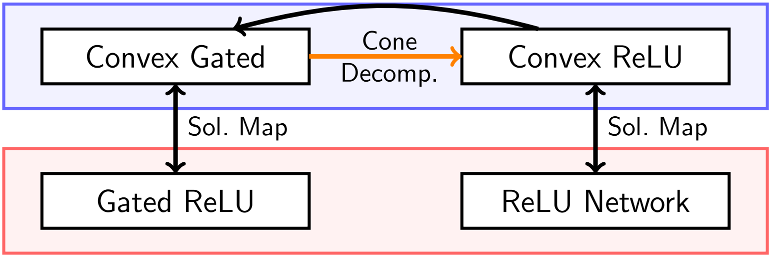
This Thesis: By creating and studying convex reformulations.



Motivation: Convexity and Deep Learning

Key Question: How can we overcome non-convexity to get better optimization and fundamental theory for neural networks?

This Thesis: By creating and studying convex reformulations.



$\hookrightarrow$ This approach yields new algorithms and new insights!

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

- They are the basic units of models like MLPs and CNNs...

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

- They are the basic units of models like MLPs and CNNs...
- ...and layer-wise training is fast and generalizes well [BEO19].

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

- They are the basic units of models like MLPs and CNNs...
- ...and layer-wise training is fast and generalizes well [BEO19].
- Two-layer networks can model self-attention [Sah+22]...

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

- They are the basic units of models like MLPs and CNNs...
- ...and layer-wise training is fast and generalizes well [BEO19].
- Two-layer networks can model self-attention [Sah+22]...
- ...and are basic components of standard transformer blocks.

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

- They are the basic units of models like MLPs and CNNs…
- …and layer-wise training is fast and generalizes well [BEO19].
- Two-layer networks can model self-attention [Sah+22]…
- …and are basic components of standard transformer blocks.
- They were key to word-embeddings (Word2Vec) [Mik+13]…

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

- They are the basic units of models like MLPs and CNNs...
- ...and layer-wise training is fast and generalizes well [BEO19].
- Two-layer networks can model self-attention [Sah+22]...
- ...and are basic components of standard transformer blocks.
- They were key to word-embeddings (Word2Vec) [Mik+13]...
- ...and are widely used in reinforcement learning [GAA23] and for prediction on edge devices [TMK17].

Motivation: Two Layer Models

We primarily study two-layer ReLU networks.

Two-layer ReLU models are **foundational building blocks**:

- They are the basic units of models like MLPs and CNNs...
- ...and layer-wise training is fast and generalizes well [BEO19].
- Two-layer networks can model self-attention [Sah+22]...
- ...and are basic components of standard transformer blocks.
- They were key to word-embeddings (Word2Vec) [Mik+13]...
- ...and are widely used in reinforcement learning [GAA23] and for prediction on edge devices [TMK17].

More Importantly: What can we achieve with two-layers?

Motivation: Tuning-Free Training Algorithms

Training neural networks involves many hyper-parameters.

Motivation: Tuning-Free Training Algorithms

Training neural networks involves many hyper-parameters.

- **Step-Size**: too small $\implies$ very slow convergence.

Motivation: Tuning-Free Training Algorithms

Training neural networks involves many hyper-parameters.

- **Step-Size**: too small $\implies$ very slow convergence.
- **Step-Size**: too big $\implies$ catastrophic failure.

Motivation: Tuning-Free Training Algorithms

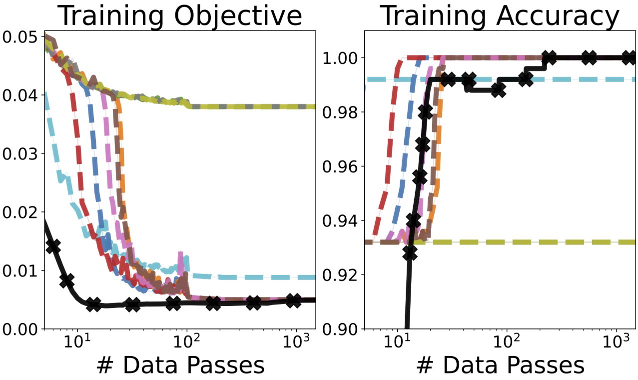
Training neural networks involves many hyper-parameters.

- **Step-Size**: too small $\implies$ very slow convergence.
- **Step-Size**: too big $\implies$ catastrophic failure.
- Not to mention batch-size, momentum, decay schedules, ...

Motivation: Tuning-Free Training Algorithms

Training neural networks involves many hyper-parameters.

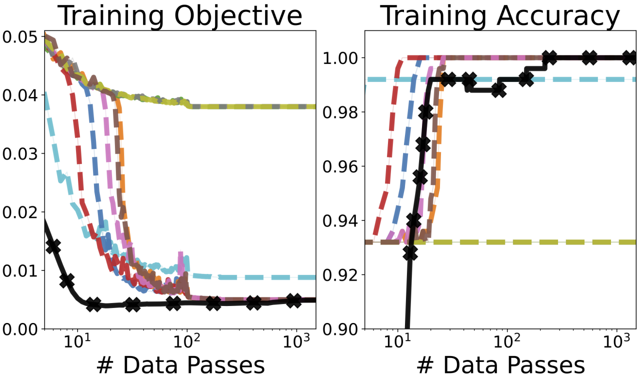
- **Step-Size**: too small $\implies$ very slow convergence.
- **Step-Size**: too big $\implies$ catastrophic failure.
- Not to mention batch-size, momentum, decay schedules, ...



Motivation: Tuning-Free Training Algorithms

Training neural networks involves many hyper-parameters.

- **Step-Size**: too small $\implies$ very slow convergence.
- **Step-Size**: too big $\implies$ catastrophic failure.
- Not to mention batch-size, momentum, decay schedules, ...



Convex reformulations enable parameter-free global optimization!

Motivation: Theory of Neural Networks

Going Beyond Optimization

- Better optimization is important for **practitioners**.

Motivation: Theory of Neural Networks

Going Beyond Optimization

- Better optimization is important for **practitioners**.
  - Less babysitting means faster, cheaper, better training.

Motivation: Theory of Neural Networks

Going Beyond Optimization

- Better optimization is important for **practitioners**.
  - Less babysitting means faster, cheaper, better training.
- But expensive hyper-parameter tuning hasn't prevented neural networks from being deployed **everywhere**.

Motivation: Theory of Neural Networks

Going Beyond Optimization

- Better optimization is important for **practitioners**.
  - Less babysitting means faster, cheaper, better training.
- But expensive hyper-parameter tuning hasn't prevented neural networks from being deployed **everywhere**.
  - They're in our cars (self-driving), phones (Face ID), classrooms (ChatGPT), and even our research (ChatGPT again)!

Motivation: Theory of Neural Networks

Going Beyond Optimization

- Better optimization is important for **practitioners**.
  - Less babysitting means faster, cheaper, better training.
- But expensive hyper-parameter tuning hasn't prevented neural networks from being deployed **everywhere**.
  - They're in our cars (self-driving), phones (Face ID), classrooms (ChatGPT), and even our research (ChatGPT again)!
- Understanding neural networks is important for **everyone**.

Motivation: Theory of Neural Networks

Going Beyond Optimization

- Better optimization is important for **practitioners**.
  - Less babysitting means faster, cheaper, better training.
- But expensive hyper-parameter tuning hasn't prevented neural networks from being deployed **everywhere**.
  - They're in our cars (self-driving), phones (Face ID), classrooms (ChatGPT), and even our research (ChatGPT again)!
- Understanding neural networks is important for **everyone**.
  - Neural network theory is necessary for safety, reliability, and future advances.

Motivation: Theory of Neural Networks

Going Beyond Optimization

- Better optimization is important for **practitioners**.
  - Less babysitting means faster, cheaper, better training.
- But expensive hyper-parameter tuning hasn't prevented neural networks from being deployed **everywhere**.
  - They're in our cars (self-driving), phones (Face ID), classrooms (ChatGPT), and even our research (ChatGPT again)!
- Understanding neural networks is important for **everyone**.
  - Neural network theory is necessary for safety, reliability, and future advances.

Let's look at some examples...

Motivation: Global Optima and Generalization

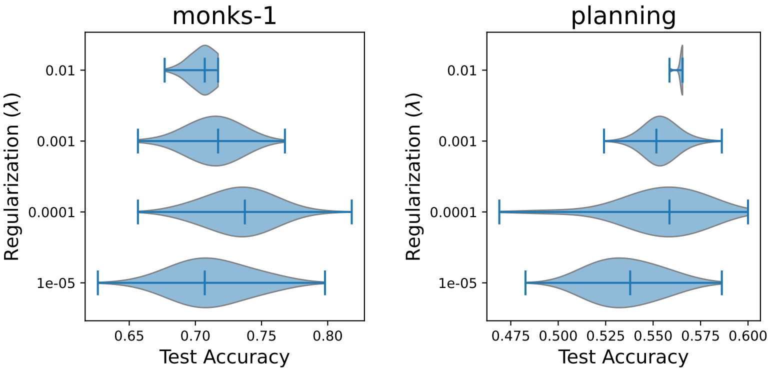
- Suppose we could take 10,000 models from the set of globally optimal neural networks given a fixed training set.

Motivation: Global Optima and Generalization

- Suppose we could take 10,000 models from the set of globally optimal neural networks given a fixed training set.
- **Q**: Are they going to generalize differently?

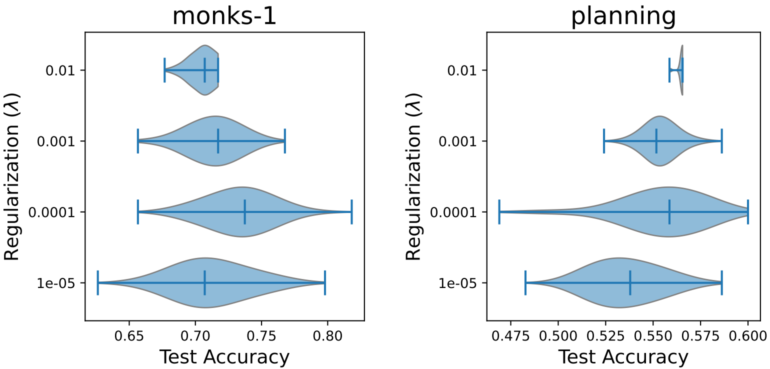
Motivation: Global Optima and Generalization

- Suppose we could take 10,000 models from the set of globally optimal neural networks given a fixed training set.
- **Q**: Are they going to generalize differently?



Motivation: Global Optima and Generalization

- Suppose we could take 10,000 models from the set of globally optimal neural networks given a fixed training set.
- **Q**: Are they going to generalize differently?



Conclusion: We need to distinguish between global optima!

Motivation: Structure of Solution Sets

Q: What is the structure of the optimal set?

Motivation: Structure of Solution Sets

Q: What is the structure of the optimal set?

- Are the solutions discrete or highly connected?

Motivation: Structure of Solution Sets

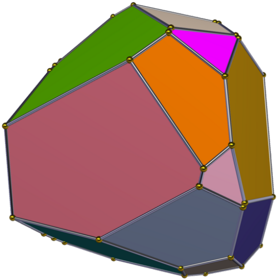
Q: What is the structure of the optimal set?

- Are the solutions discrete or highly connected?
- Are connected components disorganized or very structured?

Motivation: Structure of Solution Sets

Q: What is the structure of the optimal set?

- Are the solutions discrete or highly connected?
- Are connected components disorganized or very structured?

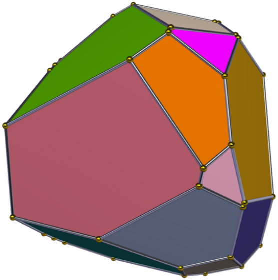


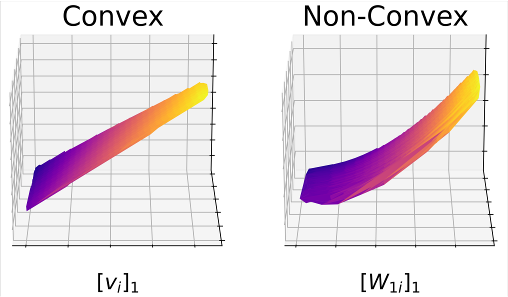
A: It's a convex polytope!

Motivation: Structure of Solution Sets

Q: What is the structure of the optimal set?

- Are the solutions discrete or highly connected?
- Are connected components disorganized or very structured?





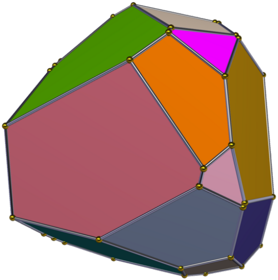
A: It's a convex polytope!

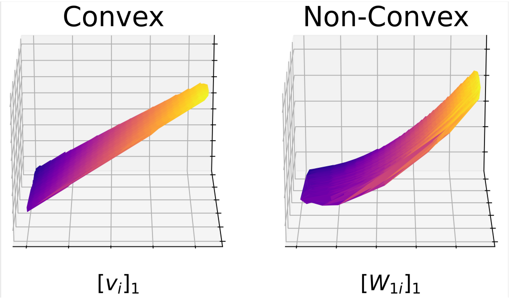
Non-Convex solution set maps the polytope to a manifold.

Motivation: Structure of Solution Sets

Q: What is the structure of the optimal set?

- Are the solutions discrete or highly connected?
- Are connected components disorganized or very structured?





A: It's a convex polytope!

Non-Convex solution set maps the polytope to a manifold.

↪ A connectivity hierarchy emerges with network width!

Overview: Big Idea

Overall Problem: neural networks are hard to train and even harder to analyze because of non-convexity.

Overview: Big Idea

Overall Problem: neural networks are hard to train and even harder to analyze because of non-convexity.

Thesis Goal: leverage **convex reformulations** of neural networks to break the barrier of non-convexity and obtain,

Overview: Big Idea

Overall Problem: neural networks are hard to train and even harder to analyze because of non-convexity.

Thesis Goal: leverage **convex reformulations** of neural networks to break the barrier of non-convexity and obtain,

- faster, more reliable optimization algorithms for training shallow neural networks;

Overview: Big Idea

Overall Problem: neural networks are hard to train and even harder to analyze because of non-convexity.

Thesis Goal: leverage **convex reformulations** of neural networks to break the barrier of non-convexity and obtain,

- faster, more reliable optimization algorithms for training shallow neural networks;
- a variational theory for the optimal set, solution path, and stability of shallow neural network optimization;

Overview: Big Idea

Overall Problem: neural networks are hard to train and even harder to analyze because of non-convexity.

Thesis Goal: leverage **convex reformulations** of neural networks to break the barrier of non-convexity and obtain,

- faster, more reliable optimization algorithms for training shallow neural networks;
- a variational theory for the optimal set, solution path, and stability of shallow neural network optimization;
- extensions to deep, fully-connected ReLU networks.

2. Background on Convex Reformulations

Convex Reformulations: Flavor of Results

Basic Idea: We start with a non-convex optimization problem and derive an equivalent convex program.

Convex Reformulations: Flavor of Results

Basic Idea: We start with a non-convex optimization problem and derive an equivalent convex program.

Equivalent means:

Convex Reformulations: Flavor of Results

Basic Idea: We start with a non-convex optimization problem and derive an equivalent convex program.

Equivalent means:

- The global minima have the same values: $p^* = q^*$

Convex Reformulations: Flavor of Results

Basic Idea: We start with a non-convex optimization problem and derive an equivalent convex program.

Equivalent means:

- The global minima have the same values: $p^* = q^*$
- We can map every global minimum u^* for one problem into a global minimum v^* of the other.

Convex Reformulations: Two-Layer ReLU Networks

Non-Convex Problem (NC-ReLU)

$$\min_{W_1, w_2} \underbrace{\frac{1}{2}\| \sum_{j=1}^{m} (XW_{1j})_+ w_{2j} - y\|_2^2}_{\text{Squared Error}} + \underbrace{\lambda \sum_{j=1}^{m} \|W_{1j}\|_2^2 + |w_{2j}|^2}_{\text{Weight Decay}},$$

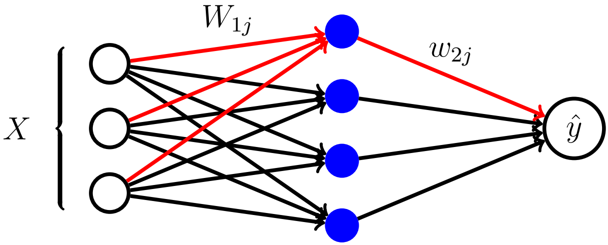
where $(z)_+ = \max\{z, 0\}$, $X \in \mathbb{R}^{n \times d}$, and $y \in \mathbb{R}^n$.

Convex Reformulations: Two-Layer ReLU Networks

Non-Convex Problem (NC-ReLU)

$$\min_{W_1, w_2} \underbrace{\frac{1}{2}\| \sum_{j=1}^{m} (XW_{1j})_+ w_{2j} - y\|_2^2}_{\text{Squared Error}} + \underbrace{\lambda \sum_{j=1}^{m} \|W_{1j}\|_2^2 + |w_{2j}|^2}_{\text{Weight Decay}},$$

where $(z)_+ = \max\{z, 0\}$, $X \in \mathbb{R}^{n \times d}$, and $y \in \mathbb{R}^n$.

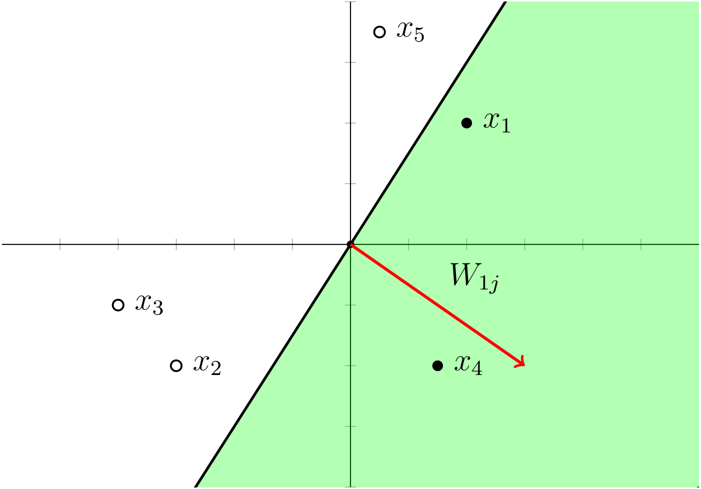


Aside: ReLU Activation Patterns

Each ReLU neuron is active on a half-space: $(XW_{1j})_+$

Aside: ReLU Activation Patterns

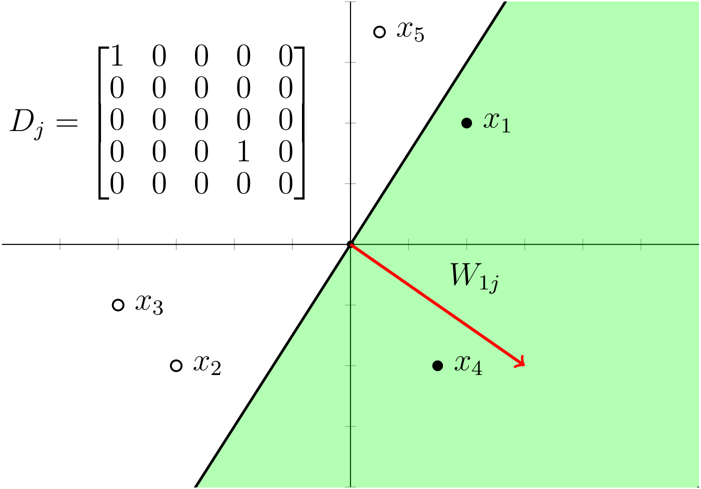
Each ReLU neuron is active on a half-space: $(XW_{1j})_+$



Aside: ReLU Activation Patterns

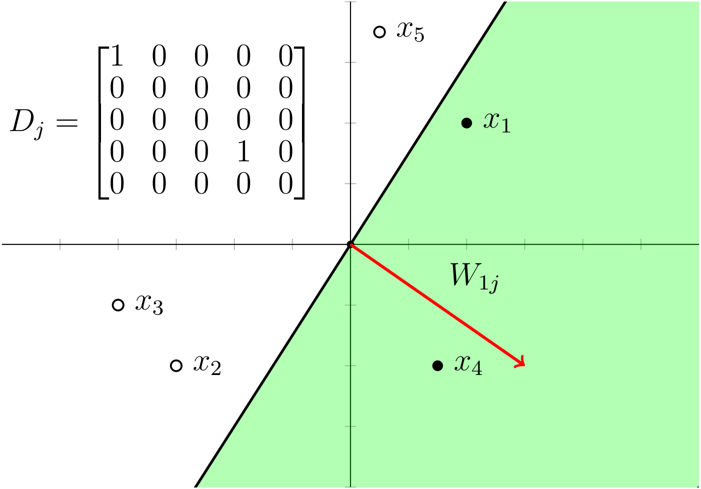
Each ReLU neuron is active on a half-space: $(XW_{1j})_+$

$$D_j = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Aside: ReLU Activation Patterns

Each ReLU neuron is active on a half-space: $(XW_{1j})_+$



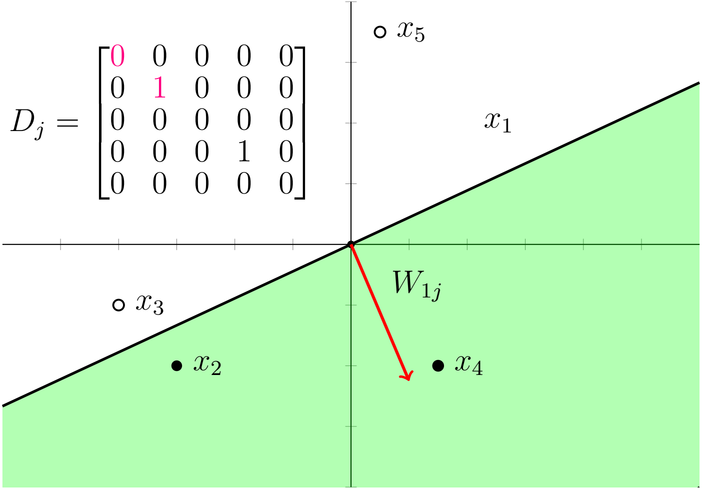
$$D_j = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Activation patterns linearize the ReLU: $(XW_{1j})_+ = D_j X W_{1j}$.

Aside: ReLU Activation Patterns

Each ReLU neuron is active on a half-space: $(XW_{1j})_+$

$$D_j = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Activation patterns linearize the ReLU: $(XW_{1j})_+ = D_j X W_{1j}$.

Convex Reformulations: Convex Problem

Convex Reformulation (C-ReLU) [PE20]

$$\min_{v,w} \frac{1}{2} \| \sum_{j=1}^{p} D_j X (v_j - w_j) - y \|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \big\{ w : \underbrace{(2D_j - I) X w \geq 0}_{\iff (Xw)_+ = D_j X w} \big\},$$

where p is the number of unique activation patterns.

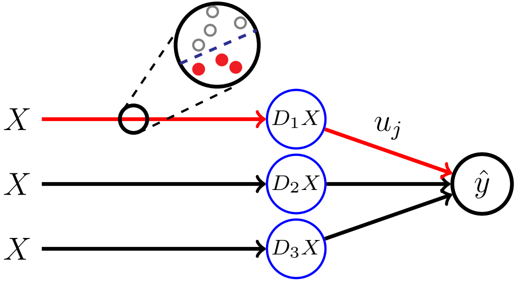
Convex Reformulations: Convex Problem

Convex Reformulation (C-ReLU) [PE20]

$$\min_{v,w} \frac{1}{2} \| \sum_{j=1}^{p} D_j X (v_j - w_j) - y \|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{ w : \underbrace{(2D_j - I)Xw \geq 0}_{\iff (Xw)_+ = D_j X w} \},$$

where p is the number of unique activation patterns.



Convex Reformulations: Hardness

Key Result: if network width m satisfies $m \geq m^*$ for some $m^* \leq n$, then C-ReLU and NC-ReLU are equivalent [PE20].

Convex Reformulations: Hardness

Key Result: if network width m satisfies $m \geq m^*$ for some $m^* \leq n$, then C-ReLU and NC-ReLU are equivalent [PE20].

How **hard** is the convex program?

Convex Reformulations: Hardness

Key Result: if network width m satisfies $m \geq m^*$ for some $m^* \leq n$, then C-ReLU and NC-ReLU are equivalent [PE20].

How **hard** is the convex program?

$$p = \left| \left\{ D_j = \mathrm{diag}[\mathbb{1}(X g_j \geq 0)] : g_j \in \mathbb{R}^d \right\} \right|$$

Convex Reformulations: Hardness

Key Result: if network width m satisfies $m \geq m^*$ for some $m^* \leq n$, then C-ReLU and NC-ReLU are equivalent [PE20].

How **hard** is the convex program?

$$p = \left| \left\{ D_j = \text{diag}[\mathbb{1}(X g_j \geq 0)] : g_j \in \mathbb{R}^d \right\} \right|$$

The **convex program** is:

- Exponential in general: $p \in O(r \cdot (\frac{n}{r})^r)$, where $r = \text{rank}(X)$.

Convex Reformulations: Hardness

Key Result: if network width m satisfies $m \geq m^*$ for some $m^* \leq n$, then C-ReLU and NC-ReLU are equivalent [PE20].

How **hard** is the convex program?

$$p = \left| \left\{ D_j = \text{diag}[\mathbb{1}(X g_j \geq 0)] : g_j \in \mathbb{R}^d \right\} \right|$$

The **convex program** is:

- Exponential in general: $p \in O(r \cdot (\frac{n}{r})^r)$, where $r = \text{rank}(X)$.
- Highly structured — it's a linear model!

Convex Reformulations: Hardness

Key Result: if network width m satisfies $m \geq m^*$ for some $m^* \leq n$, then C-ReLU and NC-ReLU are equivalent [PE20].

How **hard** is the convex program?

$$p = \left| \left\{ D_j = \text{diag}[\mathbb{1}(X g_j \geq 0)] : g_j \in \mathbb{R}^d \right\} \right|$$

The **convex program** is:

- Exponential in general: $p \in O(r \cdot (\frac{n}{r})^r)$, where $r = \text{rank}(X)$.
- Highly structured — it's a linear model!

Takeaway: We exchange one kind of hardness for another.

3. Better Optimization via Convex Reformulations

Better Optimization via Convex Reformulations

[MSP22] Fast Convex Optimization for Two-Layer ReLU Networks.
A. Mishkin, A. Sahiner, M. Pilanci. ICML 2022.

Better Optimization via Convex Reformulations

[MSP22] Fast Convex Optimization for Two-Layer ReLU Networks.
A. Mishkin, A. Sahiner, M. Pilanci. ICML 2022.

Big Idea: Use convex reformulations as an optimization tool.

Better Optimization via Convex Reformulations

[MSP22] Fast Convex Optimization for Two-Layer ReLU Networks.
A. Mishkin, A. Sahiner, M. Pilanci. ICML 2022.

Big Idea: Use convex reformulations as an optimization tool.

- We could solve C-ReLU using interior point methods, but computing the Hessian is infeasible for large n and d.

Better Optimization via Convex Reformulations

[MSP22] Fast Convex Optimization for Two-Layer ReLU Networks.
A. Mishkin, A. Sahiner, M. Pilanci. ICML 2022.

Big Idea: Use convex reformulations as an optimization tool.

- We could solve C-ReLU using interior point methods, but computing the Hessian is infeasible for large n and d.
- We could use projected GD, but projecting onto $\mathcal{K}_i$ is an expensive quadratic program.

Better Optimization via Convex Reformulations

[MSP22] Fast Convex Optimization for Two-Layer ReLU Networks.
A. Mishkin, A. Sahiner, M. Pilanci. ICML 2022.

Big Idea: Use convex reformulations as an optimization tool.

- We could solve C-ReLU using interior point methods, but computing the Hessian is infeasible for large n and d.
- We could use projected GD, but projecting onto $\mathcal{K}_i$ is an expensive quadratic program.
- Instead, we develop fast solvers based on the augmented Lagrangian method and on a "gated ReLU" relaxation.

Fast Training: Gated ReLU Networks

Recall the convex reformulation for a two-layer ReLU Network:

$$\textbf{C-ReLU}: \min_{u} \| \sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{w : (2D_j - I)Xw \geq 0\},$$

Fast Training: Gated ReLU Networks

Recall the convex reformulation for a two-layer ReLU Network:

$$\textbf{C-ReLU}: \min_{u} \|\sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \left\{w : (2D_j - I)Xw \geq 0\right\},$$

Relaxation: drop the cone constraints and simplify to obtain,

$$\textbf{C-GReLU}: \min_{u} \|\sum_{j=1}^{p} D_j X u_j - y\|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

Fast Training: Gated ReLU Networks

Recall the convex reformulation for a two-layer ReLU Network:

$$\textbf{C-ReLU}: \min_{u} \| \sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \left\{w : (2D_j - I)Xw \geq 0\right\},$$

Relaxation: drop the cone constraints and simplify to obtain,

$$\textbf{C-GReLU}: \min_{u} \| \sum_{j=1}^{p} D_j X u_j - y\|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

Questions:

Fast Training: Gated ReLU Networks

Recall the convex reformulation for a two-layer ReLU Network:

$$\textbf{C-ReLU}: \min_{u} \| \sum_{j=1}^{p} D_j X (v_j - w_j) - y \|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{ w : (2D_j - I) X w \geq 0 \},$$

Relaxation: drop the cone constraints and simplify to obtain,

$$\textbf{C-GReLU}: \min_{u} \| \sum_{j=1}^{p} D_j X u_j - y \|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

Questions:

1. Is this still a neural network?

Fast Training: Gated ReLU Networks

Recall the convex reformulation for a two-layer ReLU Network:

$$\textbf{C-ReLU}: \min_u \| \sum_{j=1}^{p} D_j X (v_j - w_j) - y \|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$
$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{ w : (2D_j - I) X w \geq 0 \},$$

Relaxation: drop the cone constraints and simplify to obtain,

$$\textbf{C-GReLU}: \min_u \| \sum_{j=1}^{p} D_j X u_j - y \|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

Questions:

1. Is this still a neural network?
2. When is it a good approximation for C-ReLU?

Fast Training: Gated ReLU Networks

1. Is CG-ReLU equivalent to a neural network architecture?

Fast Training: Gated ReLU Networks

1. Is CG-ReLU equivalent to a neural network architecture?

Theorem 2.2 [MSP22]: C-GReLU is equivalent to a neural network with a "Gated ReLU" [FMS19] activation function.

Fast Training: Gated ReLU Networks

1. Is CG-ReLU equivalent to a neural network architecture?

Theorem 2.2 [MSP22]: C-GReLU is equivalent to a neural network with a "Gated ReLU" [FMS19] activation function.

2. When does CG-ReLU give a good approximation for C-ReLU?

Fast Training: Gated ReLU Networks

1. Is CG-ReLU equivalent to a neural network architecture?

Theorem 2.2 [MSP22]: C-GReLU is equivalent to a neural network with a "Gated ReLU" [FMS19] activation function.

2. When does CG-ReLU give a good approximation for C-ReLU?

Theorem 3.7 [MSP22]: Let $\lambda \geq 0$ and let p^* be the optimal value of the ReLU problem. There exists a C-GReLU problem with minimizer u^* and optimal value d^* satisfying,

$$d^* \leq p^* \leq d^* + 2\lambda\kappa(\tilde{X}_{\mathcal{J}}) \sum_{D_i \in \tilde{\mathcal{D}}} \|u_i^*\|_2.$$

Fast Training: Solving the Convex Programs

We develop two algorithms for solving the convex reformulations:

Fast Training: Solving the Convex Programs

We develop two algorithms for solving the convex reformulations:

- **R-FISTA**: a restarted FISTA variant for Gated ReLU.

Fast Training: Solving the Convex Programs

We develop two algorithms for solving the convex reformulations:

- **R-FISTA**: a restarted FISTA variant for Gated ReLU.
- **AL**: an augmented Lagrangian method for the (constrained) ReLU Problem.

Fast Training: Solving the Convex Programs

We develop two algorithms for solving the convex reformulations:

- **R-FISTA**: a restarted FISTA variant for Gated ReLU.
- **AL**: an augmented Lagrangian method for the (constrained) ReLU Problem.

And we can use all the convex tricks!

Fast Training: Solving the Convex Programs

We develop two algorithms for solving the convex reformulations:

- **R-FISTA**: a restarted FISTA variant for Gated ReLU.
- **AL**: an augmented Lagrangian method for the (constrained) ReLU Problem.

And we can use all the convex tricks!

- **Fast**: $O(1/T^2)$ convergence rate using acceleration.

Fast Training: Solving the Convex Programs

We develop two algorithms for solving the convex reformulations:

- **R-FISTA**: a restarted FISTA variant for Gated ReLU.
- **AL**: an augmented Lagrangian method for the (constrained) ReLU Problem.

And we can use all the convex tricks!

- **Fast**: $O(1/T^2)$ convergence rate using acceleration.
- **Tuning-free**: line-search, restarts, data normalization, ...

Fast Training: Solving the Convex Programs

We develop two algorithms for solving the convex reformulations:

- **R-FISTA**: a restarted FISTA variant for Gated ReLU.
- **AL**: an augmented Lagrangian method for the (constrained) ReLU Problem.

And we can use all the convex tricks!

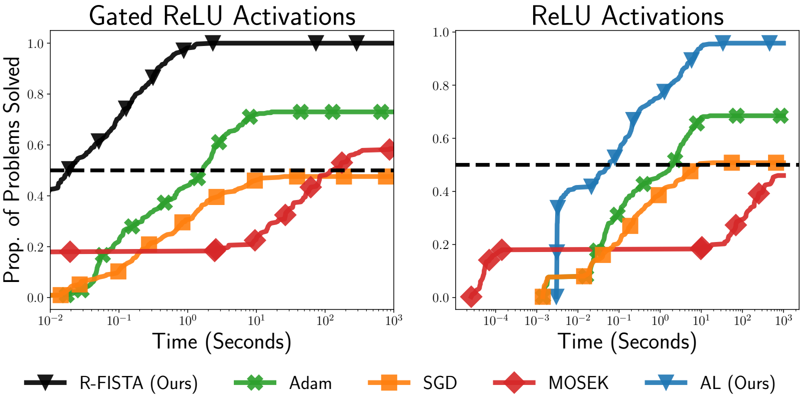
- **Fast**: $O(1/T^2)$ convergence rate using acceleration.
- **Tuning-free**: line-search, restarts, data normalization, ...
- **Certificates**: termination based on min-norm subgradient.

Fast Training: Optimization Performance

We generate a performance profile using 438 training problems from the UCI repo.

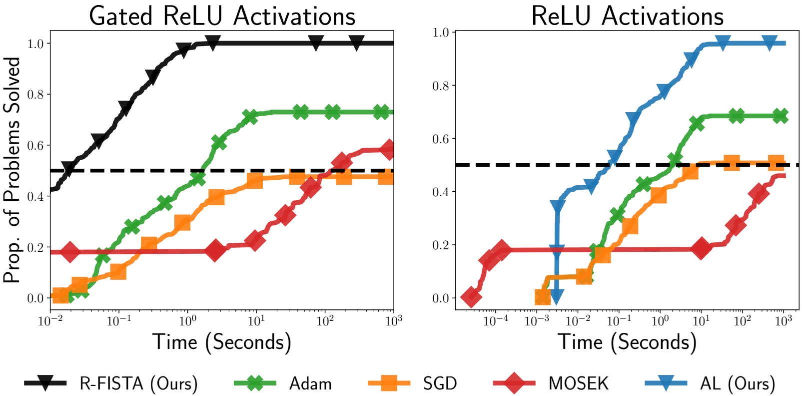
Fast Training: Optimization Performance

We generate a performance profile using 438 training problems from the UCI repo.



Fast Training: Optimization Performance

We generate a performance profile using 438 training problems from the UCI repo.



- R-FISTA/AL solve more, faster, than SGD and Adam.

4. Convex Reformulations for Theory of Neural Networks

Optimal Sets: Summary

[MP23] Optimal Sets and Solution Paths of ReLU Networks. A. Mishkin, M. Pilanci. ICML 2023

Optimal Sets: Summary

[MP23] Optimal Sets and Solution Paths of ReLU Networks. A. Mishkin, M. Pilanci. ICML 2023

Big Idea: use convex reformulations as an analytical tool to understand the set of all minimizers for two-layer ReLU networks,

Non-Convex Solution Set (NC-ReLU):

$$\mathcal{O}^*(\lambda) := \operatorname*{arg\,min}_{W_1, w_2} \; \frac{1}{2}\| \sum_{j=1}^{m} (XW_{1j})_+ w_{2j} - y\|_2^2 + \lambda \sum_{j=1}^{m} \|W_{1j}\|_2^2 + |w_{2j}|^2,$$

Optimal Set: Strong Duality

Convex Reformulation Solution Set (C-ReLU):

$$\mathcal{W}^*(\lambda) = \mathop{\arg\min}_{v_i, w_i \in \mathcal{K}_i} \left\{ \frac{1}{2} \left\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i), y \right\|_2^2 \right.$$
$$\left. + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2 \right\}.$$

Optimal Set: Strong Duality

Convex Reformulation Solution Set (C-ReLU):

$$\mathcal{W}^*(\lambda) = \mathop{\arg\min}_{v_i, w_i \in \mathcal{K}_i} \Bigg\{ \frac{1}{2} \Big\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i), y \Big\|_2^2 + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2 \Bigg\}.$$

Approach:

Optimal Set: Strong Duality

Convex Reformulation Solution Set (C-ReLU):

$$\mathcal{W}^*(\lambda) = \mathop{\arg\min}_{v_i, w_i \in \mathcal{K}_i} \left\{ \frac{1}{2} \left\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X (v_i - w_i), y \right\|_2^2 + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2 \right\}.$$

Approach:

1. Convex objective + linear constraints $\implies$ strong duality!

Optimal Set: Strong Duality

Convex Reformulation Solution Set (C-ReLU):

$$\mathcal{W}^*(\lambda) = \mathop{\arg\min}_{v_i, w_i \in \mathcal{K}_i} \left\{ \frac{1}{2} \left\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i), y \right\|_2^2 + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2 \right\}.$$

Approach:

1. Convex objective + linear constraints $\implies$ strong duality!
2. We compute the optimal set using the KKT conditions.

Optimal Set: Strong Duality

Convex Reformulation Solution Set (C-ReLU):

$$\mathcal{W}^*(\lambda) = \operatorname*{arg\,min}_{v_i, w_i \in \mathcal{K}_i} \left\{ \frac{1}{2} \left\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i), y \right\|_2^2 \right.$$
$$\left. + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2 \right\}.$$

Approach:

1. Convex objective + linear constraints $\implies$ strong duality!
2. We compute the optimal set using the KKT conditions.
3. We then map back onto the non-convex parameterization.

Optimal Set: Strong Duality

Convex Reformulation Solution Set (C-ReLU):

$$\mathcal{W}^*(\lambda) = \mathop{\arg\min}_{v_i, w_i \in \mathcal{K}_i} \left\{ \frac{1}{2} \left\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X (v_i - w_i), y \right\|_2^2 \right.$$
$$\left. + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2 \right\}.$$

Approach:

1. Convex objective + linear constraints $\implies$ strong duality!
2. We compute the optimal set using the KKT conditions.
3. We then map back onto the non-convex parameterization.
   - A little care is required to handle model symmetries.

Optimal Set: Characterization

- **Optimal Fit:** $\hat{y} := f_{\theta^*}(X) = \sum_{j=1}^{m} (XW_{1j}^*)_+ w_{2j}^*$.

Optimal Set: Characterization

- **Optimal Fit:** $\hat{y} := f_{\theta^*}(X) = \sum_{j=1}^{m}(XW_{1j}^*)_+ w_{2j}^*$.
- **Block Correlations:** A collection of unique vectors q_i, with one per activation pattern D_i.

Optimal Set: Characterization

- **Optimal Fit:** $\hat{y} := f_{\theta^*}(X) = \sum_{j=1}^{m} (XW_{1j}^*)_+ w_{2j}^*$.
- **Block Correlations:** A collection of unique vectors q_i, with one per activation pattern D_i.

Theorem 4.1 [MP23] Suppose $m \geq m^*$. Then the optimal set for NC-ReLU up to permutation/split symmetries is

$$\begin{aligned}\mathcal{O}^*(\lambda) = \big\{(W_1, w_2) :\ & f_{W_1,w_2}(X) = \hat{y}, \\ & \forall\, i \in \mathcal{S}_\lambda, W_{1i} = (\alpha_i/\lambda)^{1/2} q_i, w_{2i} = (\alpha_i \lambda)^{1/2}, \alpha_i \geq 0 \\ & \forall\, i \in [2p] \setminus \mathcal{S}_\lambda, W_{1i} = 0,\ w_{2i} = 0\big\}.\end{aligned}$$

Optimal Set: Characterization

- **Optimal Fit:** $\hat{y} := f_{\theta^*}(X) = \sum_{j=1}^{m} (XW_{1j}^*)_+ w_{2j}^*$.
- **Block Correlations:** A collection of unique vectors q_i, with one per activation pattern D_i.

Theorem 4.1 [MP23] Suppose $m \geq m^*$. Then the optimal set for NC-ReLU up to permutation/split symmetries is

$$\begin{aligned}\mathcal{O}^*(\lambda) = \big\{&(W_1, w_2) : f_{W_1,w_2}(X) = \hat{y}, \\ &\forall\, i \in \mathcal{S}_\lambda, W_{1i} = (\alpha_i/\lambda)^{1/2} q_i, w_{2i} = (\alpha_i \lambda)^{1/2}, \alpha_i \geq 0 \\ &\forall\, i \in [2p] \setminus \mathcal{S}_\lambda, W_{1i} = 0,\ w_{2i} = 0\big\}.\end{aligned}$$

- All optimal models have the same fit: $f_{W_1,w_2}(X) = \hat{y}$.

Optimal Set: Characterization

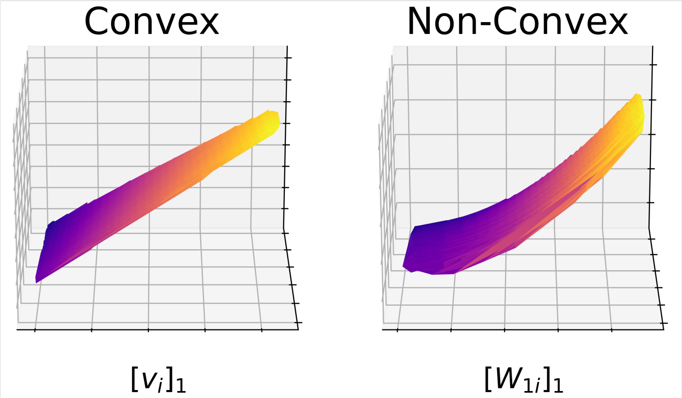
- **Optimal Fit:** $\hat{y} := f_{\theta^*}(X) = \sum_{j=1}^{m} (XW_{1j}^*)_+ w_{2j}^*$.
- **Block Correlations:** A collection of unique vectors q_i, with one per activation pattern D_i.

Theorem 4.1 [MP23] Suppose $m \geq m^*$. Then the optimal set for NC-ReLU up to permutation/split symmetries is

$$\begin{aligned} \mathcal{O}^*(\lambda) = \Big\{(W_1, w_2) :\ & f_{W_1,w_2}(X) = \hat{y}, \\ & \forall\, i \in \mathcal{S}_\lambda, W_{1i} = \left({}^{\alpha_i}/_{\lambda}\right)^{1/2} q_i, w_{2i} = \left(\alpha_i \lambda\right)^{1/2}, \alpha_i \geq 0 \\ & \forall\, i \in [2p] \setminus \mathcal{S}_\lambda, W_{1i} = 0,\ w_{2i} = 0\Big\}. \end{aligned}$$

- All optimal models have the same fit: $f_{W_1,w_2}(X) = \hat{y}$.
- The neuron directions are unique: $\frac{W_{1i}^*}{\|W_{1i}^*\|_2} = q_i/\lambda_i$.

Optimal Set: Appearance of Solutions



The non-convex parameterization maps a convex polytope of solutions into a curved manifold.

Optimal Set: Exploration and Generalization

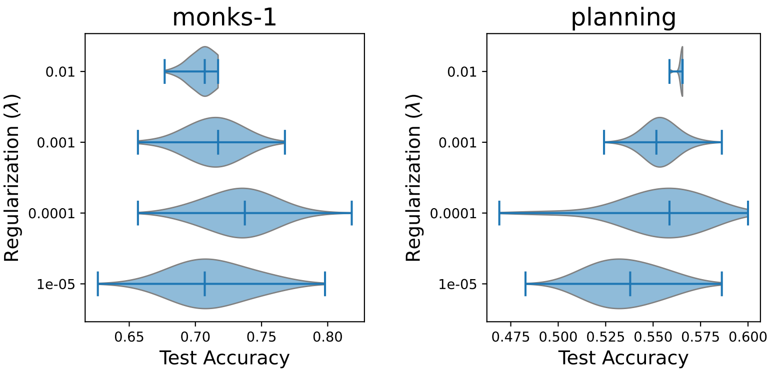
- Take 10,000 samples from the set of optimal neural networks.

Optimal Set: Exploration and Generalization

- Take 10,000 samples from the set of optimal neural networks.
- All samples have (i) **same training accuracy**, (ii) **same model norm**, but can generalize differently.

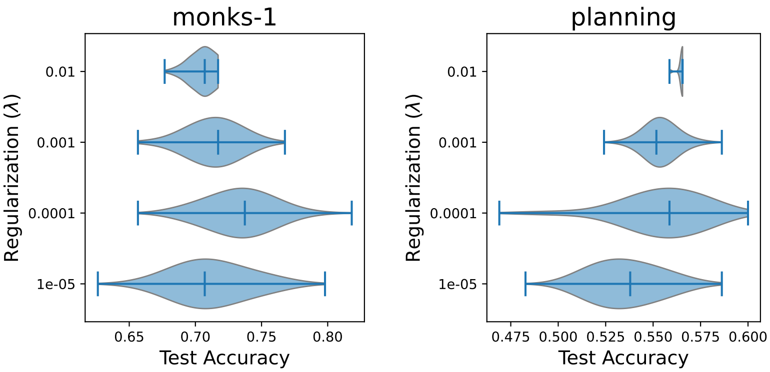
Optimal Set: Exploration and Generalization

- Take 10,000 samples from the set of optimal neural networks.
- All samples have (i) **same training accuracy**, (ii) **same model norm**, but can generalize differently.



Optimal Set: Exploration and Generalization

- Take 10,000 samples from the set of optimal neural networks.
- All samples have (i) **same training accuracy**, (ii) **same model norm**, but can generalize differently.



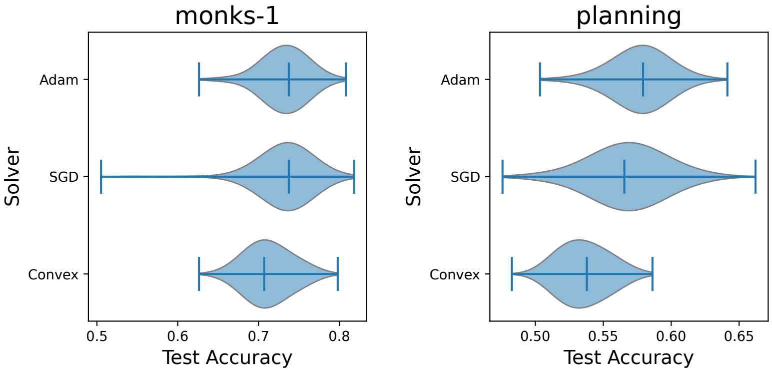
The solution you pick (implicit regularization) is crucial to good test performance!

Optimal Set: Comparison to SGD/Adam

Fix $\lambda = 10^{-5}$ and run SGD/Adam 1000 times with **independent initializations**.

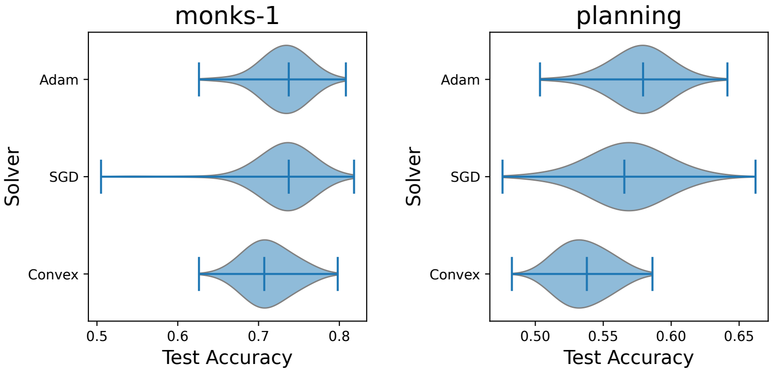
Optimal Set: Comparison to SGD/Adam

Fix $\lambda = 10^{-5}$ and run SGD/Adam 1000 times with **independent initializations**.



Optimal Set: Comparison to SGD/Adam

Fix $\lambda = 10^{-5}$ and run SGD/Adam 1000 times with **independent initializations**.



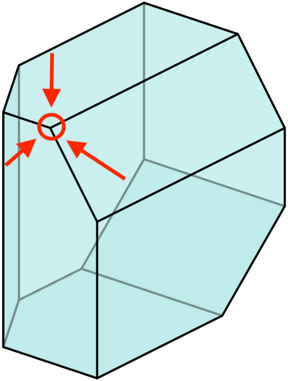
Note: SGD/Adam can converge to local minima which may perform better in this low-regularization setting.

Neuron Pruning: Minimal Models

Definition: An optimal model is minimal if there does not exist another optimal model using a strict subset of active neurons.

Neuron Pruning: Minimal Models

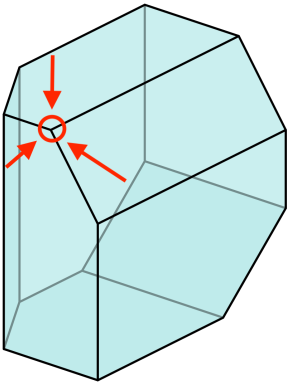
Definition: An optimal model is minimal if there does not exist another optimal model using a strict subset of active neurons.



We prove:

Neuron Pruning: Minimal Models

Definition: An optimal model is minimal if there does not exist another optimal model using a strict subset of active neurons.

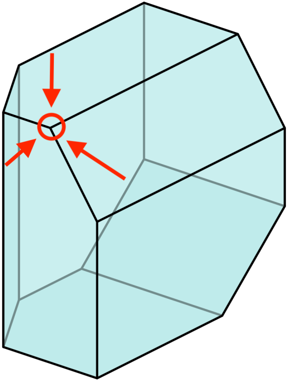


We prove:

- Vertices of the C-ReLU optimal set correspond exactly to minimal models.

Neuron Pruning: Minimal Models

Definition: An optimal model is minimal if there does not exist another optimal model using a strict subset of active neurons.

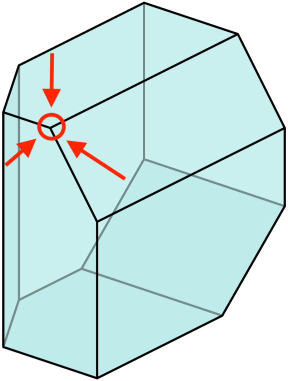


We prove:

- Vertices of the C-ReLU optimal set correspond exactly to minimal models.
- There are at most n neurons in a minimal model.

Neuron Pruning: Minimal Models

Definition: An optimal model is minimal if there does not exist another optimal model using a strict subset of active neurons.

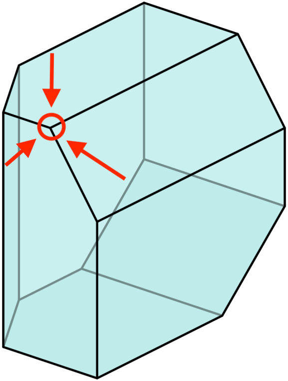


We prove:

- Vertices of the C-ReLU optimal set correspond exactly to minimal models.
- There are at most n neurons in a minimal model.
- The smallest minimal model has exactly m^* neurons.
  - $m^* \iff$ minimum width for convex reformulation.

Neuron Pruning: Minimal Models

Definition: An optimal model is minimal if there does not exist another optimal model using a strict subset of active neurons.



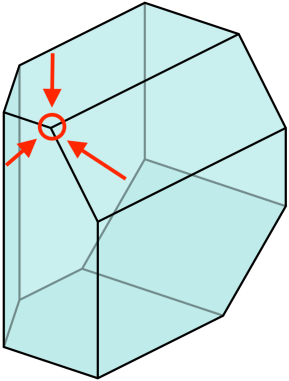
We prove:

- Vertices of the C-ReLU optimal set correspond exactly to minimal models.
- There are at most n neurons in a minimal model.
- The smallest minimal model has exactly m^* neurons.
  - $m^* \iff$ minimum width for convex reformulation.

We also give a poly-time algorithm for computing minimal models.

Neuron Pruning: Minimal Models

Definition: An optimal model is minimal if there does not exist another optimal model using a strict subset of active neurons.



We prove:

- Vertices of the C-ReLU optimal set correspond exactly to minimal models.
- There are at most n neurons in a minimal model.
- The smallest minimal model has exactly m^* neurons.
  - $m^* \iff$ minimum width for convex reformulation.

We also give a poly-time algorithm for computing minimal models.

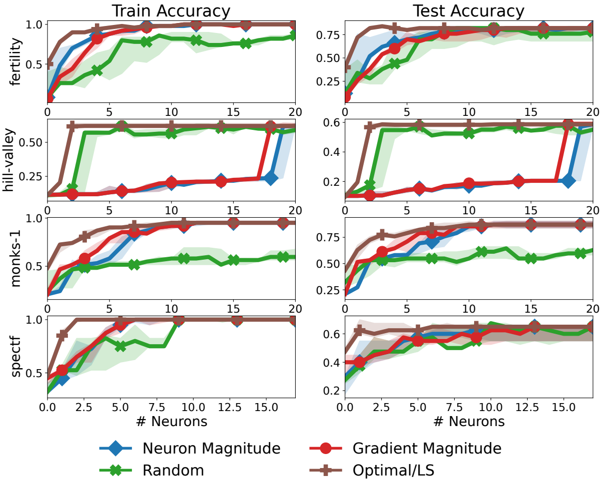
$\hookrightarrow$ This is the first optimal pruning algorithm for neural nets!

Neuron Pruning: Performance on UCI Datasets

We also show how **optimal pruning** can be adapted to prune past m^* using a simple correction step (details in bonus!).

Neuron Pruning: Performance on UCI Datasets

We also show how **optimal pruning** can be adapted to prune past m^* using a simple correction step (details in bonus!).



5. Extensions

Extensions: Scalar Inputs

[Zeg+24] A Library of Mirrors: Deep Neural Nets in Low Dimensions are Convex Lasso Models with Reflection Features. E. Zeger, Y. Wang, A. Mishkin, T. Ergen, E. Candès, M. Pilanci. SIMODS (In Review).

Extensions: Scalar Inputs

[Zeg+24] A Library of Mirrors: Deep Neural Nets in Low Dimensions are Convex Lasso Models with Reflection Features. E. Zeger, Y. Wang, A. Mishkin, T. Ergen, E. Candès, M. Pilanci. SIMODS (In Review).

Big Idea: ReLU networks with one-dimensional inputs admit a simpler convex reformulation as Lasso models.

Extensions: Scalar Inputs

[Zeg+24] A Library of Mirrors: Deep Neural Nets in Low Dimensions are Convex Lasso Models with Reflection Features. E. Zeger, Y. Wang, A. Mishkin, T. Ergen, E. Candès, M. Pilanci. SIMODS (In Review).

Big Idea: ReLU networks with one-dimensional inputs admit a simpler convex reformulation as Lasso models.

- The feature matrix for the Lasso model is determined by the model architecture.

Extensions: Scalar Inputs

[Zeg+24] A Library of Mirrors: Deep Neural Nets in Low Dimensions are Convex Lasso Models with Reflection Features. E. Zeger, Y. Wang, A. Mishkin, T. Ergen, E. Candès, M. Pilanci. SIMODS (In Review).

Big Idea: ReLU networks with one-dimensional inputs admit a simpler convex reformulation as Lasso models.

- The feature matrix for the Lasso model is determined by the model architecture.
- We extend our characterization of the set of optimal ReLU neural networks to this setting.

Extensions: Mode Connectivity

[KMP25] Exploring The Loss Landscape Of Regularized Neural Networks Via Convex Duality. S. Kim, A. Mishkin, M. Pilanci. ICLR 2025 (Oral).

Extensions: Mode Connectivity

[KMP25] Exploring The Loss Landscape Of Regularized Neural Networks Via Convex Duality. S. Kim, A. Mishkin, M. Pilanci. ICLR 2025 (Oral).

Mode Connectivity: how and when are optimal ReLU networks connected to each other in weight space?

Extensions: Mode Connectivity

[KMP25] Exploring The Loss Landscape Of Regularized Neural Networks Via Convex Duality. S. Kim, A. Mishkin, M. Pilanci. ICLR 2025 (Oral).

Mode Connectivity: how and when are optimal ReLU networks connected to each other in weight space?

- Our previous work assumed $m \geq p$: the width of the ReLU network is at least the number of activation patterns.

Extensions: Mode Connectivity

[KMP25] Exploring The Loss Landscape Of Regularized Neural Networks Via Convex Duality. S. Kim, A. Mishkin, M. Pilanci. ICLR 2025 (Oral).

Mode Connectivity: how and when are optimal ReLU networks connected to each other in weight space?

- Our previous work assumed $m \geq p$: the width of the ReLU network is at least the number of activation patterns.
- Now we study the optimal set as m ranges from m^* to p, creating a set of transitions in connectivity.

Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

1. $m = m^*$: the optimal set is a finite, fully disconnected set.

Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

1. $m = m^*$: the optimal set is a finite, fully disconnected set.
2. $m \geq m^* + 1$: there exist at least two solutions which are connected.

Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

1. $m = m^*$: the optimal set is a finite, fully disconnected set.
2. $m \geq m^* + 1$: there exist at least two solutions which are connected.
3. $m = M^*$: there exists at least one disconnected solution.

Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

1. $m = m^*$: the optimal set is a finite, fully disconnected set.
2. $m \geq m^* + 1$: there exist at least two solutions which are connected.
3. $m = M^*$: there exists at least one disconnected solution.
4. $m \geq M^* + 1$: permutations of each solution are connected. There are no disconnected points.

Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

1. $m = m^*$: the optimal set is a finite, fully disconnected set.
2. $m \geq m^* + 1$: there exist at least two solutions which are connected.
3. $m = M^*$: there exists at least one disconnected solution.
4. $m \geq M^* + 1$: permutations of each solution are connected. There are no disconnected points.
5. $m \geq \min\{n + 1, m^* + M^*\}$: the optimal set is fully connected.

Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

1. $m = m^*$: the optimal set is a finite, fully disconnected set.
2. $m \geq m^* + 1$: there exist at least two solutions which are connected.
3. $m = M^*$: there exists at least one disconnected solution.
4. $m \geq M^* + 1$: permutations of each solution are connected. There are no disconnected points.
5. $m \geq \min\{n+1, m^* + M^*\}$: the optimal set is fully connected.

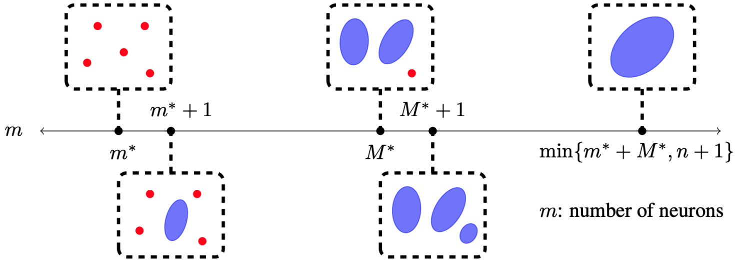
Mode Connectivity: Staircase of Connectivity

Theorem 2 [KMP25]: The critical widths m^* and M^* determine connectivity of the solution set:

1. $m = m^*$: the optimal set is a finite, fully disconnected set.
2. $m \geq m^* + 1$: there exist at least two solutions which are connected.
3. $m = M^*$: there exists at least one disconnected solution.
4. $m \geq M^* + 1$: permutations of each solution are connected. There are no disconnected points.
5. $m \geq \min\{n + 1, m^* + M^*\}$: the optimal set is fully connected.

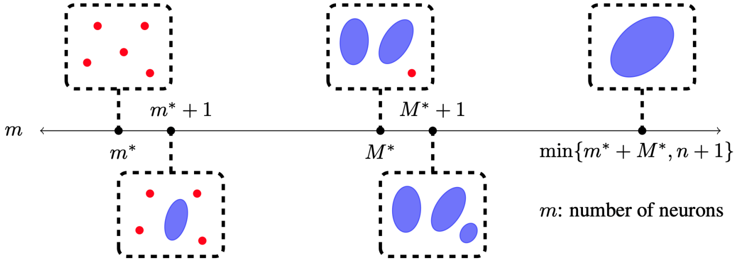
- **Credit**: this theorem is due to Sungyoon Kim, building off of my optimal set work with Mert.

Staircase of Connectivity Visualized



Mode Connectivity: Staircase in Action

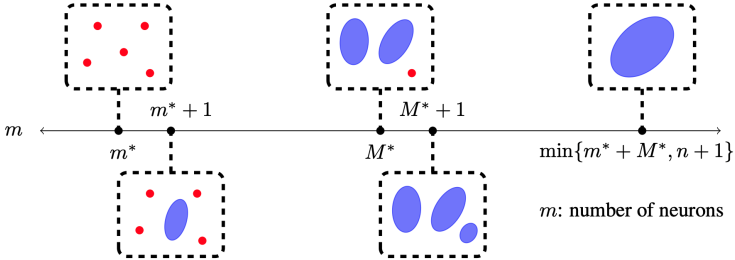
Staircase of Connectivity Visualized



Takeaway: Connectivity increases in phases with network width.

Mode Connectivity: Staircase in Action

Staircase of Connectivity Visualized



Takeaway: Connectivity increases in phases with network width.

$\hookrightarrow$ This definitively answers a long standing question in the theory of neural networks!

Extensions: Feature-Sparse Neural Networks

Convex LassoNet: Feature Sparse Convex Reformulations. A. Mishkin, T. Ergen, F. Ruan, M. Pilanci, R. Tibshirani. (Ongoing)

Extensions: Feature-Sparse Neural Networks

Convex LassoNet: Feature Sparse Convex Reformulations. A. Mishkin, T. Ergen, F. Ruan, M. Pilanci, R. Tibshirani. (Ongoing)

Big Idea: combine feature-sparsity with global optimization to improve generalization.

Extensions: Feature-Sparse Neural Networks

Convex LassoNet: Feature Sparse Convex Reformulations. A. Mishkin, T. Ergen, F. Ruan, M. Pilanci, R. Tibshirani. (Ongoing)

Big Idea: combine feature-sparsity with global optimization to improve generalization.

- Naive approaches to feature sparsity in ReLU networks using group norms underperform [FS17].

Extensions: Feature-Sparse Neural Networks

Convex LassoNet: Feature Sparse Convex Reformulations. A. Mishkin, T. Ergen, F. Ruan, M. Pilanci, R. Tibshirani. (Ongoing)

Big Idea: combine feature-sparsity with global optimization to improve generalization.

- Naive approaches to feature sparsity in ReLU networks using group norms underperform [FS17].
- But sophistical approaches like LassoNet [LRT21] are hard to train and can get trapped in local minima.

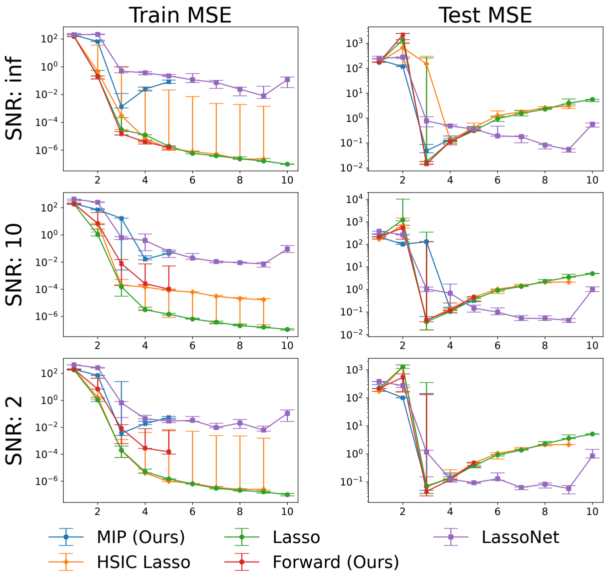
Extensions: Feature-Sparse Neural Networks

Convex LassoNet: Feature Sparse Convex Reformulations. A. Mishkin, T. Ergen, F. Ruan, M. Pilanci, R. Tibshirani. (Ongoing)

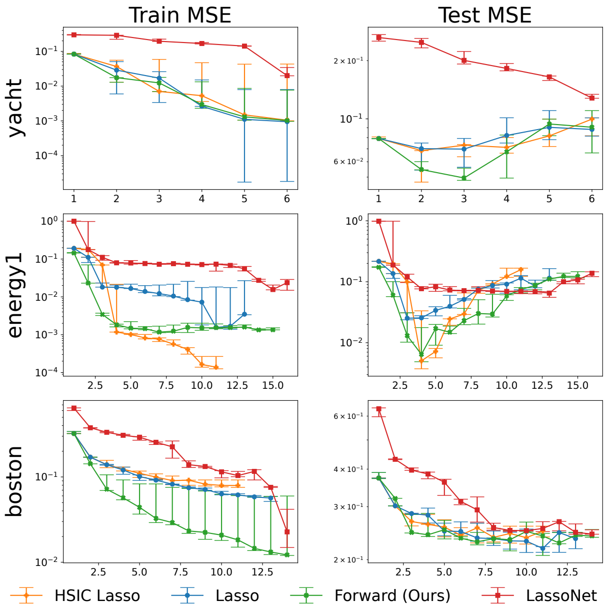
Big Idea: combine feature-sparsity with global optimization to improve generalization.

- Naive approaches to feature sparsity in ReLU networks using group norms underperform [FS17].
- But sophistical approaches like LassoNet [LRT21] are hard to train and can get trapped in local minima.
- We derive sparsity-inducing convex reformulations with global optimization guarantees.

Feature-Sparsity: Planted Neural Networks



Feature-Sparsity: Real Data



Deep Convex Reformulations: Equivalences and Optimal Sets. A. Mishkin, M. Pilanci. (Ongoing Work)

Extensions: Convexifying Deep Networks

Deep Convex Reformulations: Equivalences and Optimal Sets. A. Mishkin, M. Pilanci. (Ongoing Work)

Big Idea: Extend convex reformulations to deep ReLU networks without relying on restricted architectures.

Extensions: Convexifying Deep Networks

Deep Convex Reformulations: Equivalences and Optimal Sets. A. Mishkin, M. Pilanci. (Ongoing Work)

Big Idea: Extend convex reformulations to deep ReLU networks without relying on restricted architectures.

- Three layer networks have non-linear combinations of non-linear functions, which are challenging to analyze.

Extensions: Convexifying Deep Networks

Deep Convex Reformulations: Equivalences and Optimal Sets. A. Mishkin, M. Pilanci. (Ongoing Work)

Big Idea: Extend convex reformulations to deep ReLU networks without relying on restricted architectures.

- Three layer networks have non-linear combinations of non-linear functions, which are challenging to analyze.
- But, once we understand three layer networks, we understand k-layer networks for any $k \geq 1$.

Extensions: Convexifying Deep Networks

Deep Convex Reformulations: Equivalences and Optimal Sets. A. Mishkin, M. Pilanci. (Ongoing Work)

Big Idea: Extend convex reformulations to deep ReLU networks without relying on restricted architectures.

- Three layer networks have non-linear combinations of non-linear functions, which are challenging to analyze.
- But, once we understand three layer networks, we understand k-layer networks for any $k \geq 1$.
- We prove that ReLU MLPs of arbitrary depth are convex functions with non-convex tensor decomposition constraints.

Deep Networks: Layer Elimination

Let $W^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ and consider the k-layer ReLU network

$$f_\theta(X) = \left(\left(\left(X W^{(1)} \right)_+ W^{(2)} \right)_+ W^{(3)} \dots \right)_+ W^{(k)}.$$

Deep Networks: Layer Elimination

Let $W^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ and consider the k-layer ReLU network

$$f_\theta(X) = \left(\left(\left(XW^{(1)}\right)_+ W^{(2)}\right)_+ W^{(3)} \dots \right)_+ W^{(k)}.$$

Question: how do we construct a convex reformulation?

Deep Networks: Layer Elimination

Let $W^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ and consider the k-layer ReLU network

$$f_\theta(X) = \left(\left(\left(XW^{(1)}\right)_+ W^{(2)}\right)_+ W^{(3)} \ldots\right)_+ W^{(k)}.$$

Question: how do we construct a convex reformulation?

1. The first two-layer block has a convex reformulation in terms of the activation patterns D_i,

$$f_\theta(X) = \left(\left(\sum_{i=1}^{p} D_i X T_i^{(2)}\right)_+ \otimes W_i^{(3)} \ldots\right)_+ W^{(k)}.$$

Deep Networks: Layer Elimination

Let $W^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ and consider the k-layer ReLU network

$$f_\theta(X) = \left(\left(\left(XW^{(1)}\right)_+ W^{(2)}\right)_+ W^{(3)} \ldots\right)_+ W^{(k)}.$$

Question: how do we construct a convex reformulation?

1. The first two-layer block has a convex reformulation in terms of the activation patterns D_i,

$$f_\theta(X) = \left(\left(\sum_{i=1}^{p} D_i X T_i^{(2)}\right)_+ \otimes W_i^{(3)} \ldots\right)_+ W^{(k)}.$$

2. This creates another two-layer block, which has a convex reformulation in terms of the activation patterns $D_j^{(2)} \ldots$

Deep Networks: Tensor Programs

Let $T^{(l)} \in \mathbb{R}^{d_0 \times \dots d_l}$ be a tensor with low-rank decomposition,

$$T^{(l)} \in \mathcal{G}^{(l)} \approx \left\{ T^{(l)} : \exists\, T^{(l-1)} \in \mathcal{G}^{(l-1)} \text{ s.t. } T_i^l = T_i^{(l-1)} \otimes W_i^{(l)}. \right\}$$

Deep Networks: Tensor Programs

Let $T^{(l)} \in \mathbb{R}^{d_0 \times \ldots d_l}$ be a tensor with low-rank decomposition,

$$T^{(l)} \in \mathcal{G}^{(l)} \approx \left\{ T^{(l)} : \exists\, T^{(l-1)} \in \mathcal{G}^{(l-1)} \text{ s.t. } T_i^l = T_i^{(l-1)} \otimes W_i^{(l)}. \right\}$$

Proposition: Training a k-layer ReLU model,

$$\min_{\theta} \mathcal{L}\left(f_\theta(X), y\right),$$

is equivalent to solving the order $k+1$ tensor program

$$\min_{T^{(k)}} \mathcal{L}\left(X^{(k)} \odot T^{(k)}, y\right) \quad \text{s.t.} \quad T^{(k)} \in \mathcal{G}^{(k)}.$$

Deep Networks: Tensor Programs

Let $T^{(l)} \in \mathbb{R}^{d_0 \times \ldots d_l}$ be a tensor with low-rank decomposition,

$$T^{(l)} \in \mathcal{G}^{(l)} \approx \left\{ T^{(l)} : \exists\, T^{(l-1)} \in \mathcal{G}^{(l-1)} \text{ s.t. } T_i^l = T_i^{(l-1)} \otimes W_i^{(l)}. \right\}$$

Proposition: Training a k-layer ReLU model,

$$\min_\theta \mathcal{L}\left(f_\theta(X), y\right),$$

is equivalent to solving the order $k+1$ tensor program

$$\min_{T^{(k)}} \mathcal{L}\left(X^{(k)} \odot T^{(k)}, y\right) \quad \text{s.t.} \quad T^{(k)} \in \mathcal{G}^{(k)}.$$

- rank$(T^{(k)}) \leq d_0 \prod_{l=1}^k b_l$, where $b_l \leq d_l$ is the number of unique activation patterns occurring in layer l.

Deep Networks: Tensor Programs

Let $T^{(l)} \in \mathbb{R}^{d_0 \times \dots d_l}$ be a tensor with low-rank decomposition,

$$T^{(l)} \in \mathcal{G}^{(l)} \approx \left\{ T^{(l)} : \exists\, T^{(l-1)} \in \mathcal{G}^{(l-1)} \text{ s.t. } T_i^{l} = T_i^{(l-1)} \otimes W_i^{(l)}. \right\}$$

Proposition: Training a k-layer ReLU model,

$$\min_{\theta} \mathcal{L}\left(f_\theta(X), y\right),$$

is equivalent to solving the order $k+1$ tensor program

$$\min_{T^{(k)}} \mathcal{L}\left(X^{(k)} \odot T^{(k)}, y\right) \quad \text{s.t.} \quad T^{(k)} \in \mathcal{G}^{(k)}.$$

- rank$(T^{(k)}) \leq d_0 \prod_{l=1}^{k} b_l$, where $b_l \leq d_l$ is the number of unique activation patterns occurring in layer l.
- If $d_l \geq 2 p_l d_{l+1}$ for every l, then this becomes fully convex.

Deep Networks: Tensor Parameterization

Q: What does the tensor T represent?

Deep Networks: Tensor Parameterization

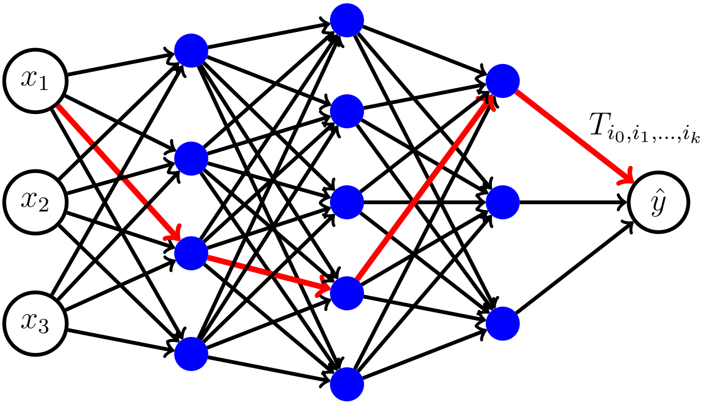
Q: What does the tensor T represent?

A: Each entry $T_{i_0,i_1,\ldots,i_k}$ is a path through the network DAG.

Deep Networks: Tensor Parameterization

Q: What does the tensor T represent?

A: Each entry $T_{i_0,i_1,\ldots,i_k}$ is a path through the network DAG.



Big-Picture Summary

Big-Picture Summary

Summary: We use convex reformulations for breakthroughs in training and neural network theory.

Big-Picture Summary

Summary: We use convex reformulations for breakthroughs in training and neural network theory.

- **Fast Training**: We develop two fast, tuning-free algorithms to train (or approximate) two-layer ReLU networks.

Big-Picture Summary

Summary: We use convex reformulations for breakthroughs in training and neural network theory.

- **Fast Training**: We develop two fast, tuning-free algorithms to train (or approximate) two-layer ReLU networks.
- **Optimal Sets**: We characterize all global optima for training two-layer ReLU networks.

Big-Picture Summary

Summary: We use convex reformulations for breakthroughs in training and neural network theory.

- **Fast Training**: We develop two fast, tuning-free algorithms to train (or approximate) two-layer ReLU networks.
- **Optimal Sets**: We characterize all global optima for training two-layer ReLU networks.
- **Deep Networks**: We show that that deep ReLU networks are tensor programs that convexify with increasing width.

Big-Picture Summary

Summary: We use convex reformulations for breakthroughs in training and neural network theory.

- **Fast Training**: We develop two fast, tuning-free algorithms to train (or approximate) two-layer ReLU networks.
- **Optimal Sets**: We characterize all global optima for training two-layer ReLU networks.
- **Deep Networks**: We show that that deep ReLU networks are tensor programs that convexify with increasing width.
- **Additional Results**: We provide many, many more results on continuity, stability, optimization, and other areas.

Overview: Publications and Ongoing Projects

Publications:

1. Fast Convex Optimization for Two-Layer ReLU Networks. A. Mishkin, A. Sahiner, M. Pilanci. ICML 2022.
2. Optimal Sets and Solution Paths of ReLU Networks. A. Mishkin, M. Pilanci. ICML 2023.
3. A Library of Mirrors: Deep Neural Nets in Low Dimensions are Convex Lasso Models with Reflection Features. E. Zeger, Y. Wang, A. Mishkin, T. Ergen, E. Candès, M. Pilanci. SIMODS (In Review).
4. Exploring The Loss Landscape Of Regularized Neural Networks Via Convex Duality. S. Kim, A. Mishkin, M. Pilanci. ICLR 2025 (Oral).

Ongoing Projects:

1. Deep Convex Reformulations: Equivalences and Optimal Sets
2. Convex LassoNet: Feature-Sparse Convex Reformulations

Overview: Additional Projects (Details in Bonus)

Additional Publications:

1. Directional Smoothness and Gradient Methods: Convergence and Adaptivity. A. Mishkin*, A. Khaled*, Y. Wang, A. Defazio, R. M. Gower. NeurIPS 2024.
2. Level Set Teleportation: An Optimization Perspective. A. Mishkin, A. Bietti, R. M. Gower. AISTATS 2025.
3. Glocal Smoothness: Line Search can really help! C. Fox, A. Mishkin, S. Vaswani, M. Schmidt. SIOPT (In Review).

Further Ongoing Projects:

1. Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm. A. V. Ramesh*, A. Mishkin*, M. Schmidt, Y. Zhou, J. Lavington, J. She. (To Be Submitted)
2. Global Convergence of Gradient Flow on 1D Data with Sign Noise. A. Mishkin, F. Bach.

Acknowledgements

My PhD has benefited from collaborations with many mentors, friends, and fellow students. Many thanks to,

Acknowledgements

My PhD has benefited from collaborations with many mentors, friends, and fellow students. Many thanks to,

- **Stanford**: Mert Pilanci, Sungyoon Kim, Tolga Ergen (LG AI), Arda Sahiner (Arcus), Emi Zeger, and many others.

Acknowledgements

My PhD has benefited from collaborations with many mentors, friends, and fellow students. Many thanks to,

- **Stanford**: Mert Pilanci, Sungyoon Kim, Tolga Ergen (LG AI), Arda Sahiner (Arcus), Emi Zeger, and many others.
- **Voleon**: Sahand Negahban, Deepak Rajan, Alex Appleton.

Acknowledgements

My PhD has benefited from collaborations with many mentors, friends, and fellow students. Many thanks to,

- **Stanford**: Mert Pilanci, Sungyoon Kim, Tolga Ergen (LG AI), Arda Sahiner (Arcus), Emi Zeger, and many others.
- **Voleon**: Sahand Negahban, Deepak Rajan, Alex Appleton.
- **Inria Paris**: Francis Bach, Frederik Kunstner.

Acknowledgements

My PhD has benefited from collaborations with many mentors, friends, and fellow students. Many thanks to,

- **Stanford**: Mert Pilanci, Sungyoon Kim, Tolga Ergen (LG AI), Arda Sahiner (Arcus), Emi Zeger, and many others.
- **Voleon**: Sahand Negahban, Deepak Rajan, Alex Appleton.
- **Inria Paris**: Francis Bach, Frederik Kunstner.
- **The Flatiron Institute**: Alberto Bietti, Robert Gower, Aaron Defazio (Meta FAIR), Ahmed Khaled (Princeton), Yuanhao Wang (Princeton).

Acknowledgements

My PhD has benefited from collaborations with many mentors, friends, and fellow students. Many thanks to,

- **Stanford**: Mert Pilanci, Sungyoon Kim, Tolga Ergen (LG AI), Arda Sahiner (Arcus), Emi Zeger, and many others.
- **Voleon**: Sahand Negahban, Deepak Rajan, Alex Appleton.
- **Inria Paris**: Francis Bach, Frederik Kunstner.
- **The Flatiron Institute**: Alberto Bietti, Robert Gower, Aaron Defazio (Meta FAIR), Ahmed Khaled (Princeton), Yuanhao Wang (Princeton).
- **UBC**: Mark Schmidt, Jonathan Lavington (Amazon), Si Yi Meng (Cornell), Sharan Vaswani (SFU).

Acknowledgements

My PhD has benefited from collaborations with many mentors, friends, and fellow students. Many thanks to,

- **Stanford**: Mert Pilanci, Sungyoon Kim, Tolga Ergen (LG AI), Arda Sahiner (Arcus), Emi Zeger, and many others.
- **Voleon**: Sahand Negahban, Deepak Rajan, Alex Appleton.
- **Inria Paris**: Francis Bach, Frederik Kunstner.
- **The Flatiron Institute**: Alberto Bietti, Robert Gower, Aaron Defazio (Meta FAIR), Ahmed Khaled (Princeton), Yuanhao Wang (Princeton).
- **UBC**: Mark Schmidt, Jonathan Lavington (Amazon), Si Yi Meng (Cornell), Sharan Vaswani (SFU).
- **RIKEN AIP**: Emtiyaz Khan, Didrik Nielsen (Twig).

Acknowledgements Continued

Lastly, I want to make special mention of the following people:

Acknowledgements Continued

Lastly, I want to make special mention of the following people:

- My committee members: Mert Pilanci, Stepehn Boyd, Mykel Kochenderfer, Balaji Prabhakar, Aaron Sidford, and Robert Tibshirani.

Acknowledgements Continued

Lastly, I want to make special mention of the following people:

- My committee members: Mert Pilanci, Stepehn Boyd, Mykel Kochenderfer, Balaji Prabhakar, Aaron Sidford, and Robert Tibshirani.
- My partner, Sophie Boerlage, for believing in me throughout this long journey.

Acknowledgements Continued

Lastly, I want to make special mention of the following people:

- My committee members: Mert Pilanci, Stepehn Boyd, Mykel Kochenderfer, Balaji Prabhakar, Aaron Sidford, and Robert Tibshirani.
- My partner, Sophie Boerlage, for believing in me throughout this long journey.
- Frederik Kunstner, without whom I probably wouldn't have stumbled into optimization.

Acknowledgements Continued

Lastly, I want to make special mention of the following people:

- My committee members: Mert Pilanci, Stepehn Boyd, Mykel Kochenderfer, Balaji Prabhakar, Aaron Sidford, and Robert Tibshirani.
- My partner, Sophie Boerlage, for believing in me throughout this long journey.
- Frederik Kunstner, without whom I probably wouldn't have stumbled into optimization.
- And, of course, my family, whose constant support made all of this possible.

References I

[AXK17] Brandon Amos, Lei Xu, and J Zico Kolter. "Input convex neural networks". In: *International conference on machine learning*. PMLR. 2017, pp. 146–155.

[Bac17] Francis Bach. "Breaking the curse of dimensionality with convex neural networks". In: *Journal of Machine Learning Research* 18.19 (2017), pp. 1–53.

[Ben+05] Yoshua Bengio et al. "Convex neural networks". In: *Advances in neural information processing systems* 18 (2005).

[BEO19] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. "Greedy Layerwise Learning Can Scale To ImageNet". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 583–593.

References II

[BT09] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[FMS19] Jonathan Fiat, Eran Malach, and Shai Shalev-Shwartz. "Decoupling gating from linearity". In: *arXiv preprint arXiv:1906.05032* (2019).

[Fox+25] Curtis Fox et al. "Glocal Smoothness: Line Search can really help!" In: *CoRR* abs/2506.12648 (2025). arXiv: 2506.12648.

[FS17] Jean Feng and Noah Simon. "Sparse-input neural networks for high-dimensional nonparametric regression and classification". In: *arXiv preprint arXiv:1711.07592* (2017).

References III

[GAA23] Mudit Gaur, Vaneet Aggarwal, and Mridul Agarwal. "On the Global Convergence of Fitted Q-Iteration with Two-layer Neural Network Parametrization". In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Vol. 202. Proceedings of Machine Learning Research. 2023, pp. 11013–11049.

[KMP25] Sungyoon Kim, Aaron Mishkin, and Mert Pilanci. "Exploring The Loss Landscape Of Regularized Neural Networks Via Convex Duality". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. 2025.

References IV

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 2012, pp. 1106–1114.

[LRT21] Ismael Lemhadri, Feng Ruan, and Robert Tibshirani. "LassoNet: Neural Networks with Feature Sparsity". In: *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*. Vol. 130. Proceedings of Machine Learning Research. 2021, pp. 10–18.

References V

[MBG25] Aaron Mishkin, Alberto Bietti, and Robert M. Gower. "Level Set Teleportation: An Optimization Perspective". In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2025, Mai Khao, Thailand, 3-5 May 2025*. Vol. 258. Proceedings of Machine Learning Research. 2025, pp. 5059–5067.

[Mik+13] Tomás Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013.

[Mis+24] Aaron Mishkin et al. "Directional Smoothness and Gradient Methods: Convergence and Adaptivity". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. 2024.

[MP23] Aaron Mishkin and Mert Pilanci. "Optimal Sets and Solution Paths of ReLU Networks". In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Vol. 202. Proceedings of Machine Learning Research. 2023, pp. 24888–24924.

[MSP22] Aaron Mishkin, Arda Sahiner, and Mert Pilanci. "Fast Convex Optimization for Two-Layer ReLU Networks: Equivalent Model Classes and Cone Decompositions". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 15770–15816.

[OC15] Brendan O'donoghue and Emmanuel Candes. "Adaptive restart for accelerated gradient schemes". In: *Foundations of computational mathematics* 15 (2015), pp. 715–732.

References VII

[PE20] Mert Pilanci and Tolga Ergen. "Neural Networks are Convex Regularizers: Exact Polynomial-time Convex Optimization Formulations for Two-layer Networks". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. 2020, pp. 7695–7705.

[PF23] Scott Pesme and Nicolas Flammarion. "Saddle-to-Saddle Dynamics in Diagonal Linear Networks". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. 2023.

[Ram+22] Aditya Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents". In: *CoRR* abs/2204.06125 (2022). arXiv: 2204.06125.

References VIII

[Ram+23] Amrutha Varshini Ramesh et al. "Analyzing and Improving Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm". In: *CoRR* abs/2307.01169 (2023). arXiv: 2307.01169.

[Rog58] Werner Wolfgang Rogosinski. "Moments of non-negative mass". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 245.1240 (1958), pp. 1–27.

[Sah+22] Arda Sahiner et al. "Unraveling Attention via Convex Duality: Analysis and Interpretations of Vision Transformers". In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Vol. 162. Proceedings of Machine Learning Research. 2022, pp. 19050–19088.

References IX

[TMK17] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. "Distributed deep neural networks over the cloud, the edge and end devices". In: *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE. 2017, pp. 328–339.

[TY09] Paul Tseng and Sangwoon Yun. "Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization". In: *Journal of optimization theory and applications* 140.3 (2009), pp. 513–535.

[Yua+21] Lu Yuan et al. "Florence: A New Foundation Model for Computer Vision". In: *CoRR* abs/2111.11432 (2021).

[Zeg+24] Emi Zeger et al. "A Library of Mirrors: Deep Neural Nets in Low Dimensions are Convex Lasso Models with Reflection Features". In: *CoRR* abs/2403.01046 (2024).

References X

[Zha+22] Bo Zhao et al. "Symmetry teleportation for accelerated optimization". In: *Advances in neural information processing systems* 35 (2022), pp. 16679–16690.

Additional Projects

Directional Smoothness: Summary

[Mis+24] Directional Smoothness and Gradient Methods: Convergence and Adaptivity. A. Mishkin*, A. Khaled*, Y. Wang, A. Defazio, R. M. Gower. NeurIPS 2024.

Directional Smoothness: Summary

[Mis+24] Directional Smoothness and Gradient Methods: Convergence and Adaptivity. A. Mishkin*, A. Khaled*, Y. Wang, A. Defazio, R. M. Gower. NeurIPS 2024.

Big Idea: GD is a local algorithm, so the analysis should be local.

Directional Smoothness: Summary

[Mis+24] Directional Smoothness and Gradient Methods: Convergence and Adaptivity. A. Mishkin*, A. Khaled*, Y. Wang, A. Defazio, R. M. Gower. NeurIPS 2024.

Big Idea: GD is a local algorithm, so the analysis should be local.

Main Contributions:

Directional Smoothness: Summary

[Mis+24] Directional Smoothness and Gradient Methods: Convergence and Adaptivity. A. Mishkin*, A. Khaled*, Y. Wang, A. Defazio, R. M. Gower. NeurIPS 2024.

Big Idea: GD is a local algorithm, so the analysis should be local.

Main Contributions:

- Directional Smoothness: concrete functions $M(x, y)$ that measure the Lipschitz smoothness of f between x and y.

Directional Smoothness: Summary

[Mis+24] Directional Smoothness and Gradient Methods: Convergence and Adaptivity. A. Mishkin*, A. Khaled*, Y. Wang, A. Defazio, R. M. Gower. NeurIPS 2024.

Big Idea: GD is a local algorithm, so the analysis should be local.

Main Contributions:

- Directional Smoothness: concrete functions $M(x, y)$ that measure the Lipschitz smoothness of f between x and y.
- Path-Dependent Rates: Convergence bounds for gradient descent that depend only on $M(x_{k+1}, x_k)$.

Directional Smoothness: Summary

[Mis+24] Directional Smoothness and Gradient Methods: Convergence and Adaptivity. A. Mishkin*, A. Khaled*, Y. Wang, A. Defazio, R. M. Gower. NeurIPS 2024.

Big Idea: GD is a local algorithm, so the analysis should be local.

Main Contributions:

- Directional Smoothness: concrete functions $M(x, y)$ that measure the Lipschitz smoothness of f between x and y.
- Path-Dependent Rates: Convergence bounds for gradient descent that depend only on $M(x_{k+1}, x_k)$.
- Adaptive Methods: algorithms which are "strongly adaptive" to the directional smoothness along the optimization path.

Level Set Teleportation: Summary

[MBG25] Level Set Teleportation: An Optimization Perspective. A. Mishkin, A. Bietti, R. M. Gower. AISTATS 2025.

Level Set Teleportation: Summary

[MBG25] Level Set Teleportation: An Optimization Perspective. A. Mishkin, A. Bietti, R. M. Gower. AISTATS 2025.

Big Idea: rigorously analyze and experimentally evaluate Newton-like "teleportation" methods [Zha+22].

Level Set Teleportation: Summary

[MBG25] Level Set Teleportation: An Optimization Perspective. A. Mishkin, A. Bietti, R. M. Gower. AISTATS 2025.

Big Idea: rigorously analyze and experimentally evaluate Newton-like "teleportation" methods [Zha+22].

Main Contributions:

Level Set Teleportation: Summary

[MBG25] Level Set Teleportation: An Optimization Perspective. A. Mishkin, A. Bietti, R. M. Gower. AISTATS 2025.

Big Idea: rigorously analyze and experimentally evaluate Newton-like "teleportation" methods [Zha+22].

Main Contributions:

- Sub-Level Set Teleportation: a co-routine to accelerate optimization by maximizing the gradient over a sub-level set,

$$x_k^+ = \arg\min_x \left\{ \|\nabla f(x)\|_2 : f(x) \leq f(x_k) \right\}.$$

Level Set Teleportation: Summary

[MBG25] Level Set Teleportation: An Optimization Perspective. A. Mishkin, A. Bietti, R. M. Gower. AISTATS 2025.

Big Idea: rigorously analyze and experimentally evaluate Newton-like "teleportation" methods [Zha+22].

Main Contributions:

- Sub-Level Set Teleportation: a co-routine to accelerate optimization by maximizing the gradient over a sub-level set,

$$x_k^+ = \arg\min_x \left\{ \|\nabla f(x)\|_2 : f(x) \le f(x_k) \right\}.$$

- We give a novel analysis combining linear progress from teleportation with sub-linear rates for non-strongly convex GD.

Level Set Teleportation: Summary

[MBG25] Level Set Teleportation: An Optimization Perspective. A. Mishkin, A. Bietti, R. M. Gower. AISTATS 2025.

Big Idea: rigorously analyze and experimentally evaluate Newton-like “teleportation” methods [Zha+22].

Main Contributions:

- Sub-Level Set Teleportation: a co-routine to accelerate optimization by maximizing the gradient over a sub-level set,

$$x_k^+ = \arg\min_x \{\|\nabla f(x)\|_2 : f(x) \leq f(x_k)\}.$$

- We give a novel analysis combining linear progress from teleportation with sub-linear rates for non-strongly convex GD.
- We also develop parameter-free algorithms for solving general, non-linear sub-level set teleportation problems in practice.

Glocal Smoothness: Summary

[Fox+25] Glocal Smoothness: Line Search can really help! C. Fox, A. Mishkin, S. Vaswani, M. Schmidt. SIOPT (In Review).

Glocal Smoothness: Summary

[Fox+25] Glocal Smoothness: Line Search can really help! C. Fox, A. Mishkin, S. Vaswani, M. Schmidt. SIOPT (In Review).

Big Question: Can we show provable speed-ups from line-search?

Glocal Smoothness: Summary

[Fox+25] Glocal Smoothness: Line Search can really help! C. Fox, A. Mishkin, S. Vaswani, M. Schmidt. SIOPT (In Review).

Big Question: Can we show provable speed-ups from line-search?

Main Contributions:

Glocal Smoothness: Summary

[Fox+25] Glocal Smoothness: Line Search can really help! C. Fox, A. Mishkin, S. Vaswani, M. Schmidt. SIOPT (In Review).

Big Question: Can we show provable speed-ups from line-search?

Main Contributions:

- Glocal smoothness: functions which are globally L-Lipschitz smooth and locally L^*-smooth $(L^* \ll L)$ around minimizers.

Glocal Smoothness: Summary

[Fox+25] Glocal Smoothness: Line Search can really help! C. Fox, A. Mishkin, S. Vaswani, M. Schmidt. SIOPT (In Review).

Big Question: Can we show provable speed-ups from line-search?

Main Contributions:

- Glocal smoothness: functions which are globally L-Lipschitz smooth and locally L^*-smooth $(L^* \ll L)$ around minimizers.
- Convergence rates showing GD with exact line-search and GD with the Polyak step-size adapt to glocal smoothness.

Glocal Smoothness: Summary

[Fox+25] Glocal Smoothness: Line Search can really help! C. Fox, A. Mishkin, S. Vaswani, M. Schmidt. SIOPT (In Review).

Big Question: Can we show provable speed-ups from line-search?

Main Contributions:

- Glocal smoothness: functions which are globally L-Lipschitz smooth and locally L^*-smooth $(L^* \ll L)$ around minimizers.
- Convergence rates showing GD with exact line-search and GD with the Polyak step-size adapt to glocal smoothness.
- Extensions to Nesterov acceleration, with and without a backtracking line-search.

Greedy Block-Coordinate Descent: Summary

[Ram+23] Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm. A. V. Ramesh*, A. Mishkin*, M. Schmidt, et al. (To Be Submitted)

Greedy Block-Coordinate Descent: Summary

[Ram+23] Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm. A. V. Ramesh*, A. Mishkin*, M. Schmidt, et al. (To Be Submitted)

Problem: greedy rules out-perform random selection for training SVMs, but rates don't reflect this.

Greedy Block-Coordinate Descent: Summary

[Ram+23] Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm. A. V. Ramesh*, A. Mishkin*, M. Schmidt, et al. (To Be Submitted)

Problem: greedy rules out-perform random selection for training SVMs, but rates don't reflect this.

Main Contributions:

Greedy Block-Coordinate Descent: Summary

[Ram+23] Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm. A. V. Ramesh*, A. Mishkin*, M. Schmidt, et al. (To Be Submitted)

Problem: greedy rules out-perform random selection for training SVMs, but rates don't reflect this.

Main Contributions:

- New analyses of block-coordinate descent for separable optimization problems with linear coupling constraints.

Greedy Block-Coordinate Descent: Summary

[Ram+23] Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm. A. V. Ramesh*, A. Mishkin*, M. Schmidt, et al. (To Be Submitted)

Problem: greedy rules out-perform random selection for training SVMs, but rates don't reflect this.

Main Contributions:

- New analyses of block-coordinate descent for separable optimization problems with linear coupling constraints.
- Single Constraint: We prove that the GS-q rule [TY09] is equivalent to steepest descent in the ℓ_1-norm, leading to fast rates with μ_1 dependence.

Greedy Block-Coordinate Descent: Summary

[Ram+23] Greedy 2-Coordinate Updates for Equality-Constrained Optimization via Steepest Descent in the 1-Norm. A. V. Ramesh*, A. Mishkin*, M. Schmidt, et al. (To Be Submitted)

Problem: greedy rules out-perform random selection for training SVMs, but rates don't reflect this.

Main Contributions:

- New analyses of block-coordinate descent for separable optimization problems with linear coupling constraints.
- Single Constraint: We prove that the GS-q rule [TY09] is equivalent to steepest descent in the ℓ_1-norm, leading to fast rates with μ_1 dependence.
- Multiple Constraints: We extend the conformal vector framework used by Tseng and Yun [TY09] to obtain rates which improve with block-size.

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Big Idea: understand how "neuron diversity" affects convergence of the gradient flow for shallow ReLU networks.

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Big Idea: understand how "neuron diversity" affects convergence of the gradient flow for shallow ReLU networks.

Main Contributions:

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Big Idea: understand how "neuron diversity" affects convergence of the gradient flow for shallow ReLU networks.

Main Contributions:

- Diversity: model neurons have different activation patterns.

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Big Idea: understand how "neuron diversity" affects convergence of the gradient flow for shallow ReLU networks.

Main Contributions:

- Diversity: model neurons have different activation patterns.
- We find diversity is preserved for simple anti-correlated noise and the gradient flow is globally convergent.

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Big Idea: understand how "neuron diversity" affects convergence of the gradient flow for shallow ReLU networks.

Main Contributions:

- Diversity: model neurons have different activation patterns.
- We find diversity is preserved for simple anti-correlated noise and the gradient flow is globally convergent.
- The gradient flow demonstrates saddle-to-saddle dynamics [PF23], with saddles corresponding to model complexity.

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Big Idea: understand how "neuron diversity" affects convergence of the gradient flow for shallow ReLU networks.

Main Contributions:

- Diversity: model neurons have different activation patterns.
- We find diversity is preserved for simple anti-correlated noise and the gradient flow is globally convergent.
- The gradient flow demonstrates saddle-to-saddle dynamics [PF23], with saddles corresponding to model complexity.

Remaining Extensions:

Gradient Flows: Summary

Global Convergence of Gradient Flow on 1D Data with Sign Noise.
A. Mishkin, F. Bach. (Ongoing)

Big Idea: understand how "neuron diversity" affects convergence of the gradient flow for shallow ReLU networks.

Main Contributions:

- Diversity: model neurons have different activation patterns.
- We find diversity is preserved for simple anti-correlated noise and the gradient flow is globally convergent.
- The gradient flow demonstrates saddle-to-saddle dynamics [PF23], with saddles corresponding to model complexity.

Remaining Extensions:

1. Generalize the target noise to more interesting/realistic regimes (e.g. fractional Brownian motion).

Background on Convex Neural Networks

Bonus: Convex Neural Networks

- Let $X \in \mathbb{R}^{n \times d}$ be a training matrix and $y \in \mathbb{R}^n$ the targets.

Bonus: Convex Neural Networks

- Let $X \in \mathbb{R}^{n \times d}$ be a training matrix and $y \in \mathbb{R}^n$ the targets.
- Let $f_\theta : \mathbb{R}^n \to \mathbb{R}$ be a neural network with parameters θ.

Bonus: Convex Neural Networks

- Let $X \in \mathbb{R}^{n \times d}$ be a training matrix and $y \in \mathbb{R}^n$ the targets.
- Let $f_\theta : \mathbb{R}^n \to \mathbb{R}$ be a neural network with parameters θ.

The standard ERM training problem is,

$$\Theta^* = \arg\min_{\theta} \sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), y_i) + r(\theta),$$

where $\mathcal{L}$ is a loss function and r is a regularizer.

Bonus: Convex Neural Networks

- Let $X \in \mathbb{R}^{n \times d}$ be a training matrix and $y \in \mathbb{R}^n$ the targets.
- Let $f_\theta : \mathbb{R}^n \to \mathbb{R}$ be a neural network with parameters θ.

The standard ERM training problem is,

$$\Theta^* = \arg\min_{\theta} \sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), y_i) + r(\theta),$$

where $\mathcal{L}$ is a loss function and r is a regularizer.

- f_θ is a convex neural network if this problem is convex in θ.

Bonus: Convex Neural Networks

- Let $X \in \mathbb{R}^{n \times d}$ be a training matrix and $y \in \mathbb{R}^n$ the targets.
- Let $f_\theta : \mathbb{R}^n \to \mathbb{R}$ be a neural network with parameters θ.

The standard ERM training problem is,

$$\Theta^* = \arg\min_\theta \sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), y_i) + r(\theta),$$

where $\mathcal{L}$ is a loss function and r is a regularizer.

- f_θ is a convex neural network if this problem is convex in θ.
- This is distinct from input-convex neural networks, where $x \mapsto f_\theta(x)$ is convex [AXK17].

Bonus: Brief Literature Review

There are multiple flavours of convex neural networks:

- Bengio et al. [Ben+05] develop a gradient-boosting algorithm where the problem of adding one neuron at a time is convex.

Bonus: Brief Literature Review

There are multiple flavours of convex neural networks:

- Bengio et al. [Ben+05] develop a gradient-boosting algorithm where the problem of adding one neuron at a time is convex.
- Bach [Bac17] study the generalization performance of infinite-width two-layer neural networks (which are convex).

Bonus: Brief Literature Review

There are multiple flavours of convex neural networks:

- Bengio et al. [Ben+05] develop a gradient-boosting algorithm where the problem of adding one neuron at a time is convex.
- Bach [Bac17] study the generalization performance of infinite-width two-layer neural networks (which are convex).
- Pilanci and Ergen [PE20] develop finite-width **convex reformulations** for two-layer ReLU networks using duality.

Bonus: Brief Literature Review

There are multiple flavours of convex neural networks:

- Bengio et al. [Ben+05] develop a gradient-boosting algorithm where the problem of adding one neuron at a time is convex.
- Bach [Bac17] study the generalization performance of infinite-width two-layer neural networks (which are convex).
- Pilanci and Ergen [PE20] develop finite-width **convex reformulations** for two-layer ReLU networks using duality.

These approaches differ primarily in how they discretize the underlying infinite-width neural network.

Bonus: Function Space Viewpoint

Bengio et al. [Ben+05] and Bach [Bac17] take a function space approach:

- Let σ be an activation function and define

$$\mathcal{H} = \left\{ h : w \in \mathbb{R}^d, h(x) = \sigma(x^\top w) \right\}.$$

Bonus: Function Space Viewpoint

Bengio et al. [Ben+05] and Bach [Bac17] take a function space approach:

- Let σ be an activation function and define

$$\mathcal{H} = \left\{ h : w \in \mathbb{R}^d, h(x) = \sigma(x^\top w) \right\}.$$

- Write problem as optimization over function space W:

$$\min_{w \in \mathcal{W}} \left\{ \sum_{j=1}^{n} L \left(\sum_{h_i \in \mathcal{H}} w_i h_i(x_j), y_j \right) + R(w) \right\}.$$

Bonus: Function Space Viewpoint

Bengio et al. [Ben+05] and Bach [Bac17] take a function space approach:

- Let σ be an activation function and define

$$\mathcal{H} = \left\{ h : w \in \mathbb{R}^d, h(x) = \sigma(x^\top w) \right\}.$$

- Write problem as optimization over function space W:

$$\min_{w \in \mathcal{W}} \left\{ \sum_{j=1}^{n} L \left(\sum_{h_i \in \mathcal{H}} w_i h_i(x_j), y_j \right) + R(w) \right\}.$$

- If R is sparsity inducing, then the final network may have finite width.

Bonus: Related Work Cont.

Bengio et al. [Ben+05]: algorithm-focused approach.

- Take $R(w) = \|w\|_1$ and $L(\hat{y}, y) = \max\{0, 1 - \hat{y}y\}$.

Bonus: Related Work Cont.

Bengio et al. [Ben+05]: algorithm-focused approach.

- Take $R(w) = \|w\|_1$ and $L(\hat{y}, y) = \max\{0, 1 - \hat{y}y\}$.
- Show that $\text{nnz}(w^*) \leq n + 1$, meaning the final model is finite.

Bonus: Related Work Cont.

Bengio et al. [Ben+05]: algorithm-focused approach.

- Take $R(w) = \|w\|_1$ and $L(\hat{y}, y) = \max\{0, 1 - \hat{y}y\}$.
- Show that $\mathrm{nnz}(w^*) \leq n + 1$, meaning the final model is finite.
- Propose a boosting-type algorithm for iteratively adding neurons.

Bonus: Related Work Cont.

Bach [Bac17]: analysis-focused approach.

- Handle spaces/functions properly using measure theory.

Bonus: Related Work Cont.

Bach [Bac17]: analysis-focused approach.

- Handle spaces/functions properly using measure theory.
  - $\mathcal{W}$ is a space of signed measures, prediction is

$$f(x) = \int_{\mathcal{H}} h(x) dw(h)$$

Bonus: Related Work Cont.

Bach [Bac17]: analysis-focused approach.

- Handle spaces/functions properly using measure theory.
  - $\mathcal{W}$ is a space of signed measures, prediction is

$$f(x) = \int_{\mathcal{H}} h(x) dw(h)$$

  - R is weighted total variation of measure w.

Bonus: Related Work Cont.

Bach [Bac17]: analysis-focused approach.

- Handle spaces/functions properly using measure theory.
  - $\mathcal{W}$ is a space of signed measures, prediction is

$$f(x) = \int_{\mathcal{H}} h(x) dw(h)$$

  - R is weighted total variation of measure w.
  - Setup reduces to Bengio et al. [Ben+05] in finite spaces.

Bonus: Related Work Cont.

Bach [Bac17]: analysis-focused approach.

- Handle spaces/functions properly using measure theory.
  - $\mathcal{W}$ is a space of signed measures, prediction is

$$f(x) = \int_{\mathcal{H}} h(x) dw(h)$$

  - R is weighted total variation of measure w.
  - Setup reduces to Bengio et al. [Ben+05] in finite spaces.
- Guarantee that $m^* \leq n$ using a representer theorem.

Bonus: Related Work Cont.

Bach [Bac17]: analysis-focused approach.

- Handle spaces/functions properly using measure theory.
  - $\mathcal{W}$ is a space of signed measures, prediction is

$$f(x) = \int_{\mathcal{H}} h(x) dw(h)$$

  - R is weighted total variation of measure w.
  - Setup reduces to Bengio et al. [Ben+05] in finite spaces.
- Guarantee that $m^* \leq n$ using a representer theorem.
- Derive an incremental algorithm based on Frank-Wolfe, but incremental steps are NP-Hard for ReLU activations.

Bonus: Key Representer Theorem

Theorem (Rogosinski [Rog58])

If $(\Omega, \mathcal{B})$ *is a Borel space,* μ *is a measure,* g_i*,* $i \in \{1, \ldots n\}$ *are measurable and* μ*-integrable, then there exists measure* $\hat{\mu}$ *with finite support at most* n *such that*

$$\int_\Omega g_i(\omega) d\mu(\omega) = \int_\Omega g_i(\omega) d\hat{\mu}(\omega)$$

for all $i \in \{1, \ldots, n\}$*.*

Prediction for dataset with n dimensions:

$$f(x_i) = \int_{\mathcal{H}} h(x_i) dw(h) = \sum_{h=1}^{m} h_j(x_i) w(h_j),$$

where $m \leq n$ and $h_j(x) = (\langle x, w_j \rangle)_+$.

Convex Reformulations

Convex Reformulations: Breaking it Down

$$\min_{u} \| \sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$
$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{w : (2D_j - I)Xw \geq 0\},$$

where $D_j = \text{diag}[\mathbb{1}(Xg_j \geq 0)]$.

- D_j is a ReLU activation pattern induced by "gate" g_j.

Convex Reformulations: Breaking it Down

$$\min_{u} \| \sum_{j=1}^{p} D_j X (v_j - w_j) - y \|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{ w : (2D_j - I) X w \geq 0 \},$$

where $D_j = \text{diag}[\mathbb{1}(X g_j \geq 0)]$.

- D_j is a ReLU activation pattern induced by "gate" g_j.
  - $[D_j]_{ii} = 1$ if $\langle x_i, g_i \rangle \geq 0$ and 0 otherwise.

Convex Reformulations: Breaking it Down

$$\min_{u} \| \sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{w : (2D_j - I)Xw \geq 0\}$$

where $D_j = \text{diag}[\mathbb{1}(Xg_j \geq 0)]$.

- D_j is a ReLU activation pattern induced by "gate" g_j.
  - $[D_j]_{ii} = 1$ if $\langle x_i, g_i \rangle \geq 0$ and 0 otherwise.
- Weight-decay regularization turns into "group ℓ_1" penalty.

Convex Reformulations: Breaking it Down

$$\min_u \| \sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{w : (2D_j - I)Xw \geq 0\},$$

where $D_j = \text{diag}[\mathbb{1}(Xg_j \geq 0)]$.

- D_j is a ReLU activation pattern induced by "gate" g_j.
  - $[D_j]_{ii} = 1$ if $\langle x_i, g_i \rangle \geq 0$ and 0 otherwise.
- Weight-decay regularization turns into "group ℓ_1" penalty.
- The constraint $v_j \in \mathcal{K}_j$ implies

$$(Xv_j)_+ = D_j X v_j.$$

That is, v_j has the activation encoded by D_j.

Bonus: Explicit Solution Mapping

Given (v^*, w^*), an optimal non-convex ReLU network is given by

C to NC:
$$W_{1i} = v_i^* / \sqrt{\|v_i^*\|}, \quad w_{2i} = \sqrt{\|v_i^*\|}$$
$$W_{1j} = w_i^* / \sqrt{\|w_i^*\|}, \quad w_{2j} = -\sqrt{\|w_i^*\|}.$$

- Optimal solution balances weight between layers.

Given (W_{1i}^*, w_{2i}^*), an optimal convex ReLU model is

NC to C:
$$v_i = W_{1i}^* \left| w_{2i} \right|^* \qquad \text{if } w_{2i}^* \geq 0$$
$$w_i = W_{1i}^* \left| w_{2i} \right|^* \qquad \text{Otherwise.}$$

- Optimal solution combines weight from both layers.

Gated ReLU Networks and Cone Decompositions

Bonus: Gated ReLU Networks

Theorem 2.2 (informal): C-GReLU is equivalent to solving

$$\textbf{NC-GReLU}: \min_{W_1, \alpha} \frac{1}{2} \| \sum_{j=1}^{p} \phi_{g_j}(X, w_j)\alpha - y\|_2^2 + \frac{\lambda}{2} \sum_{j=1}^{p} \|w_j\|_2^2 + |\alpha_j|^2,$$

with the "Gated ReLU" [FMS19] activation function

$$\phi_g(X, u) = \mathsf{diag}(\mathbb{1}(Xg \geq 0))Xu,$$

and gate vectors g_j such that

$$D_j = \mathsf{diag}[\mathbb{1}(Xg_j \geq 0).$$

Bonus: Gated ReLU Networks

Theorem 2.2 (informal): C-GReLU is equivalent to solving

NC-GReLU : $\min_{W_1,\alpha} \frac{1}{2} \| \sum_{j=1}^{p} \phi_{g_j}(X, w_j)\alpha - y\|_2^2 + \frac{\lambda}{2} \sum_{j=1}^{p} \|w_j\|_2^2 + |\alpha_j|^2,$

with the "Gated ReLU" [FMS19] activation function

$$\phi_g(X, u) = \mathrm{diag}(\mathbb{1}(Xg \geq 0))Xu,$$

and gate vectors g_j such that

$$D_j = \mathrm{diag}[\mathbb{1}(Xg_j \geq 0).$$

Interpretation: if $u_j \notin \mathcal{K}_j$, then the activation must be decoupled from the linear mapping in the non-convex model.

Bonus: Cone Decompositions

Question: when are Gated ReLU and ReLU networks equivalent?

Question: when are Gated ReLU and ReLU networks equivalent?

Consider special case where $\lambda = 0$.

$$\textbf{C-GReLU} : \min_u \| \sum_{j=1}^{p} D_j X u_j - y \|_2^2.$$

V.S.

$$\textbf{C-ReLU} : \min_u \| \sum_{j=1}^{p} D_j X (v_j - w_j) - y \|_2^2.$$

$$\text{s.t. } v_j, w_j \in \mathcal{K}_j := \{ w : (2D_j - I) X w \geq 0 \},$$

Bonus: Equivalent Statement

Equiv. Question: when does $u_j = v_j - w_j$ for some $v_j, w_j \in \mathcal{K}_j$?

Bonus: Equivalent Statement

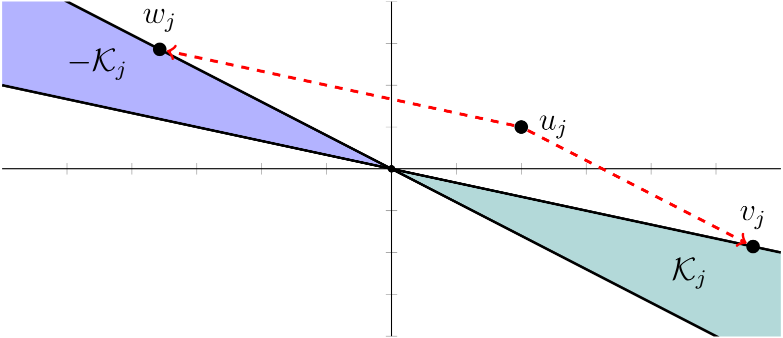
Equiv. Question: when does $u_j = v_j - w_j$ for some $v_j, w_j \in \mathcal{K}_j$?

Answer: when $\mathcal{K}_j - \mathcal{K}_j = \mathbb{R}^d$ and a "cone decomposition" exists.

Bonus: Equivalent Statement

Equiv. Question: when does $u_j = v_j - w_j$ for some $v_j, w_j \in \mathcal{K}_j$?

Answer: when $\mathcal{K}_j - \mathcal{K}_j = \mathbb{R}^d$ and a "cone decomposition" exists.



Bonus: Basic Cone Decomposition

Recall: $\mathcal{K}_j = \{w : (2D_j - I)Xw \geq 0\}$.

Bonus: Basic Cone Decomposition

Recall: $\mathcal{K}_j = \{w : (2D_j - I)Xw \geq 0\}$.

- This is a polyhedral cone which we rewrite as

$$\mathcal{K}_j = \bigcap_{i=1}^{n} \{w : [S_j]_{ii} \cdot \langle x_i, w \rangle \geq 0\},$$

where $S_j = (2D_j - I)$.

Bonus: Basic Cone Decomposition

Recall: $\mathcal{K}_j = \{w : (2D_j - I)Xw \geq 0\}$.

- This is a polyhedral cone which we rewrite as

$$\mathcal{K}_j = \bigcap_{i=1}^{n} \{w : [S_j]_{ii} \cdot \langle x_i, w \rangle \geq 0\},$$

where $S_j = (2D_j - I)$.

Proposition 3.1 (informal): If X is full row-rank, then $\text{aff}(\mathcal{K}_j) = \mathbb{R}^d$ and $\mathcal{K}_j - \mathcal{K}_j = \mathbb{R}^d$.

Bonus: Basic Cone Decomposition

Recall: $\mathcal{K}_j = \{w : (2D_j - I)Xw \geq 0\}$.

- This is a polyhedral cone which we rewrite as

$$\mathcal{K}_j = \bigcap_{i=1}^{n} \{w : [S_j]_{ii} \cdot \langle x_i, w \rangle \geq 0\},$$

where $S_j = (2D_j - I)$.

Proposition 3.1 (informal): If X is full row-rank, then $\text{aff}(\mathcal{K}_j) = \mathbb{R}^d$ and $\mathcal{K}_j - \mathcal{K}_j = \mathbb{R}^d$.

Unfortunately, there is no extension to full-rank X.

Bonus: Not All Cones are Equal

Alternative Idea: show we don't need "singular" cones $\mathcal{K}_j$,

$$\mathcal{K}_j - \mathcal{K}_j \subsetneq \mathbb{R}^d.$$

Bonus: Not All Cones are Equal

Alternative Idea: show we don't need "singular" cones $\mathcal{K}_j$,

$$\mathcal{K}_j - \mathcal{K}_j \subsetneq \mathbb{R}^d.$$

Proposition 3.2 (informal): Suppose $\mathcal{K}_j - \mathcal{K}_j \subset \mathbb{R}^d$. Then, there exists $\mathcal{K}_i$ for which $\mathcal{K}_i - \mathcal{K}_i = \mathbb{R}^d$ and $\mathcal{K}_j \subset \mathcal{K}_i$.

Bonus: Not All Cones are Equal

Alternative Idea: show we don't need "singular" cones $\mathcal{K}_j$,

$$\mathcal{K}_j - \mathcal{K}_j \subsetneq \mathbb{R}^d.$$

Proposition 3.2 (informal): Suppose $\mathcal{K}_j - \mathcal{K}_j \subset \mathbb{R}^d$. Then, there exists $\mathcal{K}_i$ for which $\mathcal{K}_i - \mathcal{K}_i = \mathbb{R}^d$ and $\mathcal{K}_j \subset \mathcal{K}_i$.

Interpretation: if optimal $u_j^* \neq 0$, then set

$$u_i' = u_j^* + u_i^*.$$

It is possible to show this causes no problems.

Bonus: Cone Decomposition Proof Sketch

Proof: Works by iteratively constructing $\mathcal{K}_i$ s.t. $\mathcal{K}_j \subset \mathcal{K}_i$.

Bonus: Cone Decomposition Proof Sketch

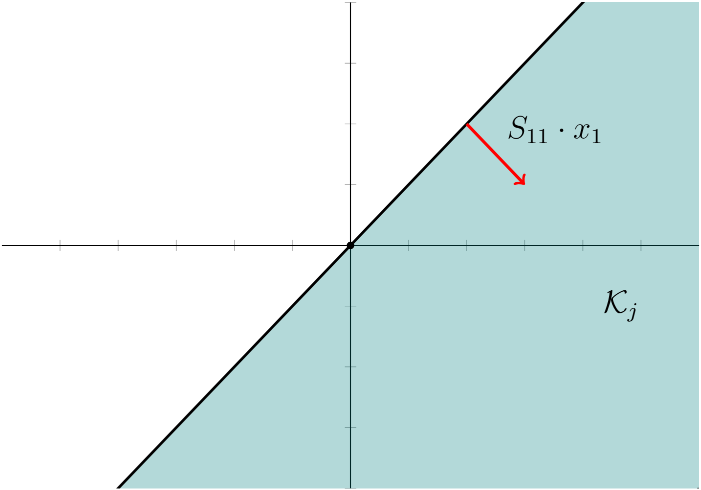
Proof: Works by iteratively constructing $\mathcal{K}_i$ s.t. $\mathcal{K}_j \subset \mathcal{K}_i$.

We sketch a simpler statement:

Proposition 3.2 (informal): Suppose $\mathcal{K}_j = \{0\}$. Then, there exists $\mathcal{K}_i$ for which $\mathcal{K}_i - \mathcal{K}_i = \mathbb{R}^d$ and $\mathcal{K}_j \subset \mathcal{K}_i$.

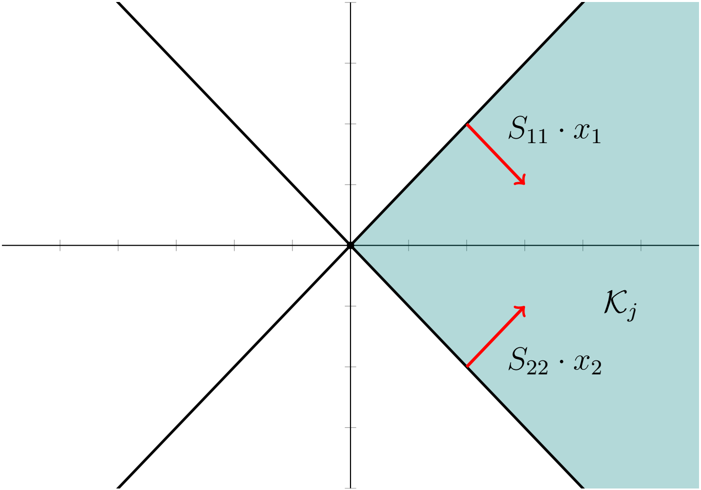
Bonus: Cone Decomposition Proof Sketch

$$\mathcal{K}'_j = \{w : [S_j]_{11} \cdot \langle x_1, w \rangle \geq 0\}$$



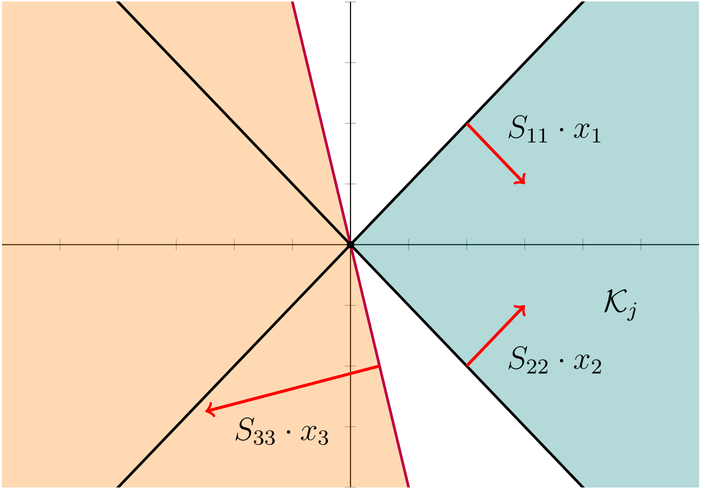
Cone Decompositions: Proof Sketch

$$\mathcal{K}_j'' = \mathcal{K}_j' \cap \{w : [S_j]_{22} \cdot \langle x_2, w \rangle \geq 0\}$$



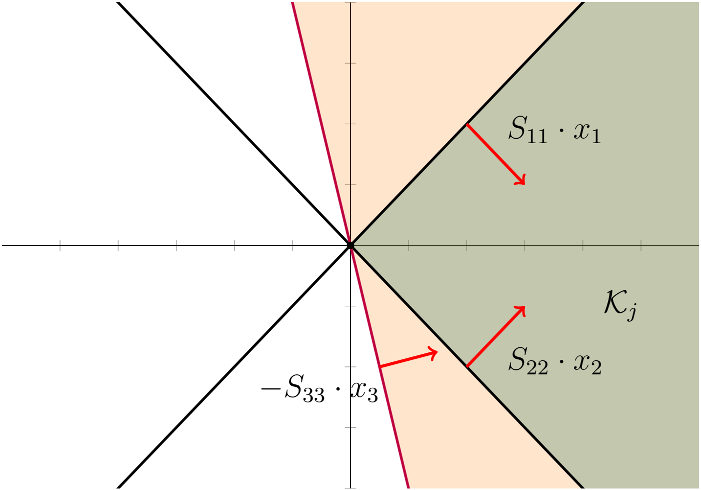
Bonus: Cone Decomposition Proof Sketch

$$\mathcal{K}_j''' = \mathcal{K}_j'' \cap \{w : [S_j]_{33} \cdot \langle x_3, w \rangle \geq 0\}$$



Bonus: Cone Decomposition Proof Sketch

$$\tilde{\mathcal{K}}_j''' = \mathcal{K}_j'' \cap \{w : -[S_j]_{33} \cdot \langle x_3, w \rangle \geq 0\}$$



Bonus: Main Cone Decomposition Result

- The real proof is more complex, but this is the core idea.
  - Build $\mathcal{K}_i$ by switching signs of $[S_j]_{ii}$.
  - Equivalent to turning on/off activations.
- Leads to our main approximation result.

Bonus: Main Cone Decomposition Result

- The real proof is more complex, but this is the core idea.
  - Build $\mathcal{K}_i$ by switching signs of $[S_j]_{ii}$.
  - Equivalent to turning on/off activations.
- Leads to our main approximation result.

Theorem 3.7 (informal): Let $\lambda \geq 0$ and let p^* be the optimal value of the ReLU problem. There exists a C-GReLU problem with minimizer u^* and optimal value d^* satisfying,

$$d^* \leq p^* \leq d^* + 2\lambda\kappa(\tilde{X}_{\mathcal{J}}) \sum_{D_i \in \tilde{\mathcal{D}}} \|u_i^*\|_2.$$

Details on Optimization Algorithms

Bonus: ReLU by Cone Decomposition

1. Solve the gated ReLU problem:

$$u^* \in \arg\min_{u} \| \sum_{j=1}^{p} D_j X u_j - y\|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

Bonus: ReLU by Cone Decomposition

1. Solve the gated ReLU problem:

$$u^* \in \arg\min_u \|\sum_{j=1}^{p} D_j X u_j - y\|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

2. Solve a cone decomposition:

$$v_j^*, w_j^* \in \arg\min_{v_j, w_j} \left\{ L(v_j, w_j) : v_j - w_j = u_j^* \right\},$$

where L is a loss function.

Bonus: ReLU by Cone Decomposition

1. Solve the gated ReLU problem:

$$u^* \in \arg\min_{u} \| \sum_{j=1}^{p} D_j X u_j - y\|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

2. Solve a cone decomposition:

$$v_j^*, w_j^* \in \arg\min_{v_j, w_j} \left\{ L(v_j, w_j) : v_j - w_j = u_j^* \right\},$$

   where L is a loss function.

3. Compute corresponding ReLU model.

Bonus: ReLU by Cone Decomposition

1. Solve the gated ReLU problem:

$$u^* \in \arg\min_u \| \sum_{j=1}^{p} D_j X u_j - y\|_2^2 + \lambda \sum_{j=1}^{p} \|u_j\|_2$$

2. Solve a cone decomposition:

$$v_j^*, w_j^* \in \arg\min_{v_j, w_j} \left\{ L(v_j, w_j) : v_j - w_j = u_j^* \right\},$$

   where L is a loss function.

3. Compute corresponding ReLU model.

Choosing:

- $L(v, w) = \|v\|_2 + \|w\|_2$ gives an SOCP.

Bonus: ReLU by Cone Decomposition

1. Solve the gated ReLU problem:

$$u^* \in \arg\min_u \| \sum_{j=1}^p D_j X u_j - y\|_2^2 + \lambda \sum_{j=1}^p \|u_j\|_2$$

2. Solve a cone decomposition:

$$v_j^*, w_j^* \in \arg\min_{v_j, w_j} \left\{ L(v_j, w_j) : v_j - w_j = u_j^* \right\},$$

   where L is a loss function.

3. Compute corresponding ReLU model.

Choosing:

- $L(v, w) = \|v\|_2 + \|w\|_2$ gives an SOCP.
- $L(v, w) = 0$ yields a linear feasibility problem.

Bonus: R-FISTA

Consider "composite" optimization problem:

$$\min_x f(x) + g(x),$$

where f is L-smooth and g is convex. Smoothness implies

$$\begin{aligned} f(y) &\le Q_{x_k,1/L}(y) \\ &= f(x_k) + \langle \nabla f(x_k), y - x_k \rangle + \frac{L}{2}\|y - x_k\|_2^2. \end{aligned}$$

Bonus: R-FISTA

Consider "composite" optimization problem:

$$\min_x f(x) + g(x),$$

where f is L-smooth and g is convex. Smoothness implies

$$\begin{aligned} f(y) &\leq Q_{x_k,1/L}(y) \\ &= f(x_k) + \langle \nabla f(x_k), y - x_k \rangle + \frac{L}{2} \|y - x_k\|_2^2. \end{aligned}$$

The **FISTA** algorithm minimizes Q_{y_k,η_k} and handles g exactly:

$$\begin{aligned} x_{k+1} &= \arg\min_y Q_{y_k,\eta_k}(y) + g(y) \\ y_{k+1} &= x_{k+1} + \frac{t_k - 1}{t_{k+1}} (x_{k+1} - x_k) \end{aligned}$$

where $t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2$.

Bonus: R-FISTA Continued

We combine this with line-search and restarts:

Bonus: R-FISTA Continued

We combine this with line-search and restarts:

- **Line-search**: backtrack on η_k until:

$$f(x_{k+1}(\eta_k)) \leq Q_{y_k,\eta_k}(x_{k+1}(\eta_k)),$$

  as proposed by [BT09].

Bonus: R-FISTA Continued

We combine this with line-search and restarts:

- **Line-search**: backtrack on η_k until:

  $$f(x_{k+1}(\eta_k)) \leq Q_{y_k,\eta_k}(x_{k+1}(\eta_k)),$$

  as proposed by [BT09].

- **Restarts**: reset to $y_k = x_k$ if

  $$\langle x_{k+1} - x_k, x_{k+1} - y_k \rangle > 0,$$

  that is, x_{k+1} is not a descent step with respect to proximal-gradient mapping [OC15].

Bonus: R-FISTA Continued

We combine this with line-search and restarts:

- **Line-search**: backtrack on η_k until:

  $$f(x_{k+1}(\eta_k)) \leq Q_{y_k,\eta_k}(x_{k+1}(\eta_k)),$$

  as proposed by [BT09].

- **Restarts**: reset to $y_k = x_k$ if

  $$\langle x_{k+1} - x_k, x_{k+1} - y_k \rangle > 0,$$

  that is, x_{k+1} is not a descent step with respect to proximal-gradient mapping [OC15].

- And lots of other **convex tricks**...

Bonus: AL Method

Let $\tilde{X}_i = (2D_i - I)X$ so that $v_j \in \mathcal{K}_j \iff X_j v_j \geq 0$ and define

$$F(v, w) = \| \sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2.$$

Now we can form the augmented Lagrangian:

$$\begin{aligned} \mathcal{L}_\delta(v,w,\gamma,\zeta) := (\delta/2) \sum_{D_i \in \tilde{\mathcal{D}}} \big[\|(\gamma_i/\delta - \tilde{X}_i v_i)_+\|_2^2 \\ + \|(\zeta_i/\delta - \tilde{X}_i w_i)_+\|_2^2 \big] + F(v, w). \end{aligned} \tag{1}$$

We use the multiplier method to update the dual parameters:

$$\begin{aligned} (v_{k+1}, w_{k+1}) &= \arg\min_{v,w} \mathcal{L}_\delta(v, w, \gamma_k, \zeta_k), \\ \gamma_{k+1} = (\gamma_k - \delta \tilde{X}_i v_i)_+, &\quad \zeta_{k+1} = (\zeta_k - \delta \tilde{X}_i w_i)_+. \end{aligned}$$

Bonus: AL Method

Let $\tilde{X}_i = (2D_i - I)X$ so that $v_j \in \mathcal{K}_j \iff X_j v_j \geq 0$ and define

$$F(v, w) = \| \sum_{j=1}^{p} D_j X(v_j - w_j) - y\|_2^2 + \lambda \sum_{j=1}^{p} \|v_j\|_2 + \|w_j\|_2.$$

Now we can form the augmented Lagrangian:

$$\begin{aligned}\mathcal{L}_\delta(v,w,\gamma,\zeta) := (\delta/2) \sum_{D_i \in \tilde{\mathcal{D}}} \Big[\|(\gamma_i/\delta - \tilde{X}_i v_i)_+\|_2^2 \\ + \|(\zeta_i/\delta - \tilde{X}_i w_i)_+\|_2^2 \Big] + F(v, w).\end{aligned} \tag{1}$$

We use the multiplier method to update the dual parameters:

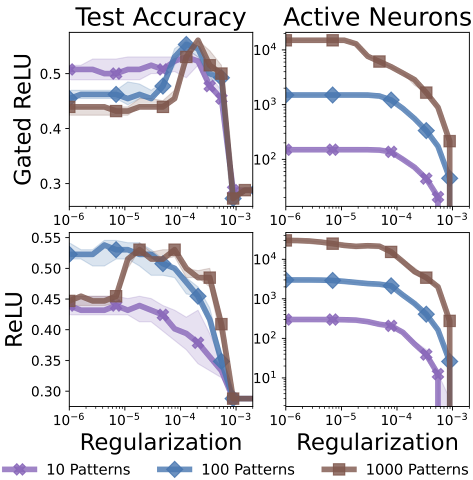
$$(v_{k+1}, w_{k+1}) = \arg\min_{v,w} \mathcal{L}_\delta(v, w, \gamma_k, \zeta_k),$$

$$\gamma_{k+1} = (\gamma_k - \delta \tilde{X}_i v_i)_+, \quad \zeta_{k+1} = (\zeta_k - \delta \tilde{X}_i w_i)_+.$$

We use warm starts and propose a heuristic for δ.

Additional Optimization Experiments

Bonus: Sub-sampling Patterns



- Variance induced by resampling $\tilde{\mathcal{D}}$ is minimal.
- Standard bias-variance trade-off.

Bonus: Generalization Performance

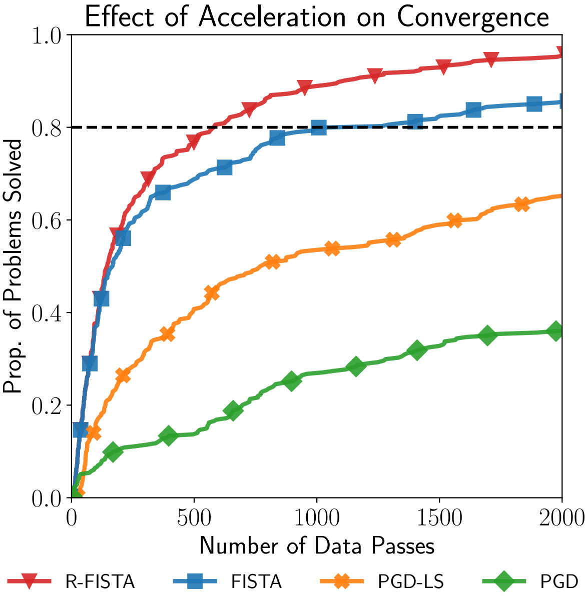
Generalization performance is equivalent to non-convex solvers.

Dataset	Convex	Adam	SGD
magic	85.9	**86.9**	86.4
statlog-heart	**83.3**	**83.3**	79.6
vertebral-col.	**90.3**	**90.3**	88.7
cardiotocogr.	**89.9**	36.5	88.9
abalone	**66.2**	65.3	66.1
annealing	90.6	**93.7**	88.7
car	87.8	**94.8**	90.1
bank	89.8	**90.8**	90.5
breast-cancer	**68.4**	64.9	**68.4**
page-blocks	94.0	**97.1**	96.9
contrac	**55.1**	54.4	53.7
congressional	63.2	62.1	**67.8**
spambase	93.3	**93.5**	93.2
synthetic	**98.3**	96.7	96.7
hill-valley	**65.3**	62.8	55.4

Bonus: Comparison to Standard Baselines

Dataset	**C-GReLU**	**C-ReLU**	**RF**	**Linear**	**RBF**
blood	79.9	**80.5**	75.8	74.5	77.9
chess-krvkp	**99.2**	98.6	98.9	97.2	98.4
conn-bench	**90.2**	85.4	73.2	68.3	85.4
cylinder-bands	76.5	**78.4**	77.5	71.6	71.6
fertility	**80.0**	**80.0**	75.0	75.0	75.0
heart-hung.	**86.2**	**86.2**	84.5	84.5	**86.2**
hill-valley	**76.0**	68.6	57.9	62.0	70.2
ilpd-liver	72.4	**74.1**	66.4	71.6	71.6
mammographic	77.6	78.6	**80.7**	**80.7**	80.2
monks-1	**100**	**100**	95.8	79.2	83.3
musk-1	94.7	**95.8**	92.6	86.3	**95.8**
ozone	**97.6**	**97.6**	97.4	97.2	97.4
pima	74.5	74.5	**76.5**	75.2	73.2
planning	**69.4**	63.9	66.7	66.7	**69.4**
spambase	93.5	93.6	**94.1**	92.2	93.6
spectf	**87.5**	75.0	68.8	68.8	68.8
statlog-german	74.0	**77.5**	73.5	75.0	75.5
tic-tac-toe	99.0	99.0	99.5	98.4	**100**

Bonus: Acceleration Ablation



Example: Discontinuous Regularization Paths

Example: Discontinuous Paths

Consider training a toy neural network: given $(x_1, y_1), (x_2, y_2)$,

Example: Discontinuous Paths

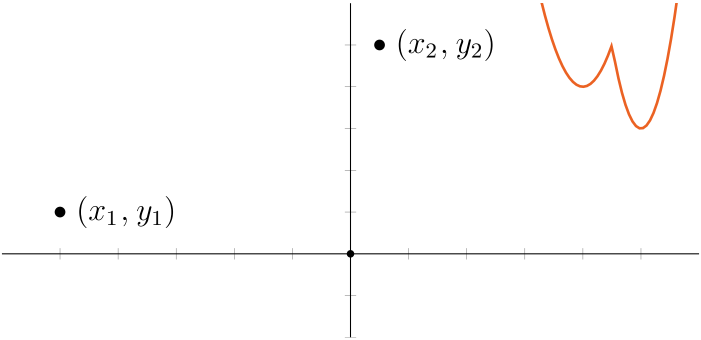
Consider training a toy neural network: given $(x_1, y_1), (x_2, y_2)$,

$$\min_{w_1} \frac{1}{2}((w_1 x_1)_+ - y_1)^2 + \frac{1}{2}((w_1 x_2)_+ - y_2)^2 + \lambda |w_1|.$$

Example: Discontinuous Paths

Consider training a toy neural network: given $(x_1, y_1), (x_2, y_2)$,

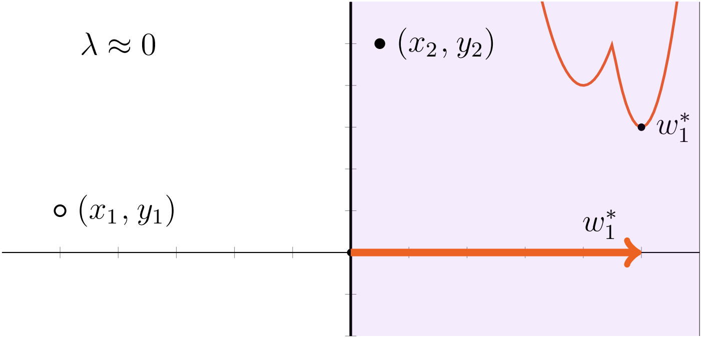
$$\min_{w_1} \frac{1}{2}((w_1 x_1)_+ - y_1)^2 + \frac{1}{2}((w_1 x_2)_+ - y_2)^2 + \lambda |w_1|.$$



Example: Discontinuous Paths

Consider training a toy ReLU network:

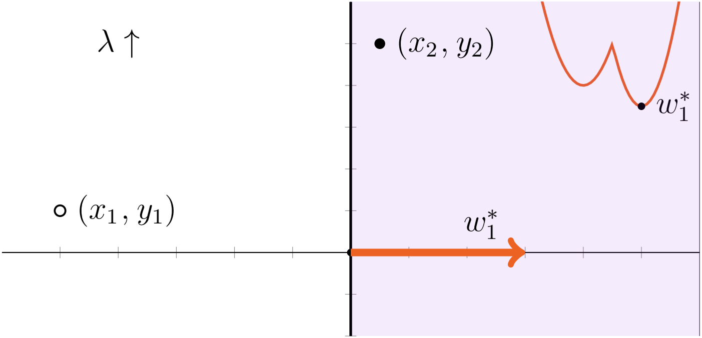
$$\min_{w_1} \frac{1}{2}((w_1 x_1)_+ - y_1)^2 + \frac{1}{2}((w_1 x_2)_+ - y_2)^2 + \lambda |w_1|.$$



Example: Discontinuous Paths

Consider training a toy neural network:

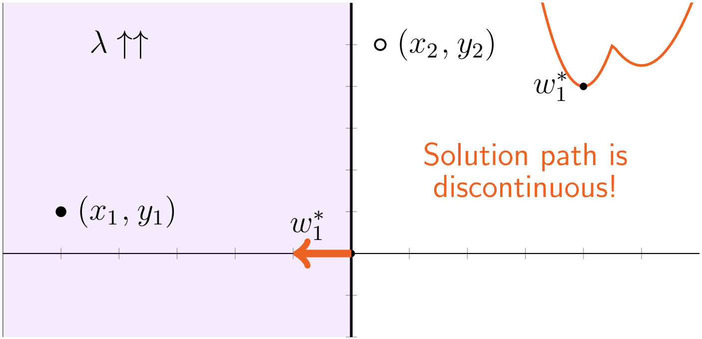
$$\min_{w_1} \frac{1}{2}((w_1 x_1)_+ - y_1)^2 + \frac{1}{2}((w_1 x_2)_+ - y_2)^2 + \lambda|w_1|.$$



Example: Discontinuous Paths

Consider training a toy neural network:

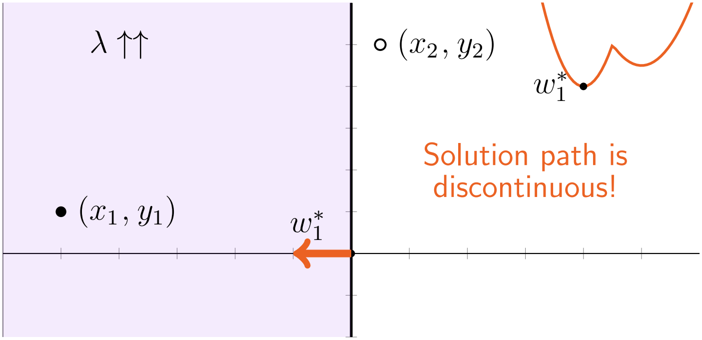
$$\min_{w_1} \frac{1}{2}((w_1 x_1)_+ - y_1)^2 + \frac{1}{2}((w_1 x_2)_+ - y_2)^2 + \lambda |w_1|.$$



Example: Discontinuous Paths

Consider training a toy neural network:

$$\min_{w_1} \frac{1}{2}((w_1 x_1)_+ - y_1)^2 + \frac{1}{2}((w_1 x_2)_+ - y_2)^2 + \lambda |w_1|.$$



Goal: Overcome these problems via convexification.

Optimal Sets

Bonus: C-ReLU Optimality Conditions

We form the Lagrangian for the convex reformulation:

$$\mathcal{L}(v, w, \rho^+, \rho^-) = \frac{1}{2}\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i) - y\|_2^2 + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2$$
$$- \sum_{D_i \tilde{\mathcal{D}}} \left[\left\langle \tilde{X}_i^\top \rho^-, w \right\rangle - \left\langle \tilde{X}_i^\top \rho^+, v \right\rangle \right],$$

where $\tilde{X}_i = (2D_i - I)$.

Bonus: C-ReLU Optimality Conditions

We form the Lagrangian for the convex reformulation:

$$\mathcal{L}(v, w, \rho^+, \rho^-) = \frac{1}{2}\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i) - y\|_2^2 + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2 \\ - \sum_{D_i \tilde{\mathcal{D}}} \left[\left\langle \tilde{X}_i^\top \rho^-, w \right\rangle - \left\langle \tilde{X}_i^\top \rho^+, v \right\rangle \right],$$

where $\tilde{X}_i = (2D_i - I)$.

The KKT conditions are necessary and sufficient for optimality:

Bonus: C-ReLU Optimality Conditions

We form the Lagrangian for the convex reformulation:

$$\mathcal{L}(v, w, \rho^+, \rho^-) = \frac{1}{2}\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i) - y\|_2^2 + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2$$
$$- \sum_{D_i \tilde{\mathcal{D}}} \left[\left\langle \tilde{X}_i^\top \rho^-, w \right\rangle - \left\langle \tilde{X}_i^\top \rho^+, v \right\rangle \right],$$

where $\tilde{X}_i = (2D_i - I)$.

The KKT conditions are necessary and sufficient for optimality:

- Stationary Lagrangian:

$$\underbrace{X^\top D_i(\hat{y} - y) - \tilde{X}_i^\top \rho_i^+}_{q_i^+} \in \partial \lambda \|v_i\|_2.$$

Bonus: C-ReLU Optimality Conditions

We form the Lagrangian for the convex reformulation:

$$\mathcal{L}(v, w, \rho^+, \rho^-) = \frac{1}{2}\| \sum_{D_i \in \tilde{\mathcal{D}}} D_i X(v_i - w_i) - y\|_2^2 + \lambda \sum_{D_i \in \tilde{\mathcal{D}}} \|v_i\|_2 + \|w_i\|_2$$
$$- \sum_{D_i \tilde{\mathcal{D}}} \left[\left\langle \tilde{X}_i{}^\top \rho^-, w \right\rangle - \left\langle \tilde{X}_i{}^\top \rho^+, v \right\rangle \right],$$

where $\tilde{X}_i = (2D_i - I)$.

The KKT conditions are necessary and sufficient for optimality:

- Stationary Lagrangian:

$$\underbrace{X^\top D_i(\hat{y} - y) - \tilde{X}_i{}^\top \rho_i^+}_{q_i^+} \in \partial \lambda \|v_i\|_2.$$

  - It turns out each q_i^+ is **unique** WLOG!

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

- Model fit $\hat{y}$ is constant over optimal set $\mathcal{W}^*(\lambda)$.

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

- Model fit $\hat{y}$ is constant over optimal set $\mathcal{W}^*(\lambda)$.
- Implies correlation $X^\top D_i(y - \hat{y})$ is constant over $\mathcal{W}^*(\lambda)$.

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

- Model fit $\hat{y}$ is constant over optimal set $\mathcal{W}^*(\lambda)$.
- Implies correlation $X^\top D_i(y - \hat{y})$ is constant over $\mathcal{W}^*(\lambda)$.
- We may assume ρ is unique (e.g. min-norm dual solution).

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

- Model fit $\hat{y}$ is constant over optimal set $\mathcal{W}^*(\lambda)$.
- Implies correlation $X^\top D_i(y - \hat{y})$ is constant over $\mathcal{W}^*(\lambda)$.
- We may assume ρ is unique (e.g. min-norm dual solution).

Non-zero Blocks:

- Suppose $\theta_i \neq 0$.

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

- Model fit $\hat{y}$ is constant over optimal set $\mathcal{W}^*(\lambda)$.
- Implies correlation $X^\top D_i(y - \hat{y})$ is constant over $\mathcal{W}^*(\lambda)$.
- We may assume ρ is unique (e.g. min-norm dual solution).

Non-zero Blocks:

- Suppose $\theta_i \neq 0$.
- Then $\nabla \|\theta_i\|_2 = s_i = \lambda \theta_i / \|\theta_i\|_2$.

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

- Model fit $\hat{y}$ is constant over optimal set $\mathcal{W}^*(\lambda)$.
- Implies correlation $X^\top D_i(y - \hat{y})$ is constant over $\mathcal{W}^*(\lambda)$.
- We may assume ρ is unique (e.g. min-norm dual solution).

Non-zero Blocks:

- Suppose $\theta_i \neq 0$.
- Then $\nabla \|\theta_i\|_2 = s_i = \lambda \theta_i / \|\theta_i\|_2$.
- Rearranging stationarity implies $\exists \alpha_i > 0$:

$$\theta_i = \alpha_i \underbrace{\left[X^\top D_i(y - \hat{y}) - \tilde{X}_i \rho_i \right]}_{q_i}.$$

Bonus: Characterizing the Optimal Set

Facts: let (θ, ρ) be primal dual optimal.

- Model fit $\hat{y}$ is constant over optimal set $\mathcal{W}^*(\lambda)$.
- Implies correlation $X^\top D_i(y - \hat{y})$ is constant over $\mathcal{W}^*(\lambda)$.
- We may assume ρ is unique (e.g. min-norm dual solution).

Non-zero Blocks:

- Suppose $\theta_i \neq 0$.
- Then $\nabla \|\theta_i\|_2 = s_i = \lambda \theta_i / \|\theta_i\|_2$.
- Rearranging stationarity implies $\exists \alpha_i > 0$:

$$\theta_i = \alpha_i \underbrace{\left[X^\top D_i(y - \hat{y}) - \tilde{X}_i \rho_i \right]}_{q_i}.$$

- Every solution is a non-negative multiple of these q_i vectors.

Bonus: Explicit Optimal Set

We gave a characterization of $\mathcal{W}^*(\lambda)$ that depends on

$$\mathcal{S}_\lambda = \{i \in [2p] : \exists \theta \in \mathcal{W}^*(\lambda),\ \theta_i \neq 0\}.$$

Alternative expression involves additional linear constraints.

Bonus: Explicit Optimal Set

We gave a characterization of $\mathcal{W}^*(\lambda)$ that depends on

$$\mathcal{S}_\lambda = \{i \in [2p] : \exists \theta \in \mathcal{W}^*(\lambda),\ \theta_i \neq 0\}.$$

Alternative expression involves additional linear constraints.

$$\begin{aligned}\mathcal{W}^*(\lambda) = \big\{\theta : \forall\, i \in \mathcal{E}_\lambda, \theta_i = \alpha_i q_i, \alpha_i \geq 0, \\ \forall\, j \in [2p] \setminus \mathcal{E}_\lambda, \theta_j = 0, \sum_{i=1}^{2p} D_i X \theta_i = \hat{y}, \\ \forall\, i \in [2p], \tilde{X}_i \theta_i \geq 0, \left\langle \rho, \tilde{X}_i \theta_i \right\rangle = 0. \big\}\end{aligned}$$

Bonus: Explicit Optimal Set

We gave a characterization of $\mathcal{W}^*(\lambda)$ that depends on

$$\mathcal{S}_\lambda = \{i \in [2p] : \exists \theta \in \mathcal{W}^*(\lambda),\ \theta_i \neq 0\}.$$

Alternative expression involves additional linear constraints.

$$\begin{aligned}\mathcal{W}^*(\lambda) = \big\{\theta : \forall\, i \in \mathcal{E}_\lambda, \theta_i = \alpha_i q_i, \alpha_i \geq 0, \\ \forall\, j \in [2p] \setminus \mathcal{E}_\lambda, \theta_j = 0, \sum_{i=1}^{2p} D_i X \theta_i = \hat{y}, \\ \forall\, i \in [2p], \tilde{X}_i \theta_i \geq 0, \left\langle \rho, \tilde{X}_i \theta_i \right\rangle = 0. \big\}\end{aligned}$$

More complex, but also **explicit**.

Bonus: Solution Mapping for C-ReLU

Given (v^*, w^*), an optimal non-convex ReLU network is given by

C to NC:
$$W_{1i} = v_i^* / \sqrt{\|v_i^*\|}, \quad w_{2i} = \sqrt{\|v_i^*\|}$$
$$W_{1j} = w_i^* / \sqrt{\|w_i^*\|}, \quad w_{2j} = -\sqrt{\|w_i^*\|}.$$

Bonus: Solution Mapping for C-ReLU

Given (v^*, w^*), an optimal non-convex ReLU network is given by

C to NC:

$$W_{1i} = v_i^* / \sqrt{\|v_i^*\|}, \quad w_{2i} = \sqrt{\|v_i^*\|}$$

$$W_{1j} = w_i^* / \sqrt{\|w_i^*\|}, \quad w_{2j} = -\sqrt{\|w_i^*\|}.$$

- Optimal convex weights satisfy $v_i^* = \alpha_i q_i$ so that

$$\|v_i^*\|_2 = \alpha_i \|q_i\|_2 = \alpha_i \lambda.$$

Bonus: Solution Mapping for C-ReLU

Given (v^*, w^*), an optimal non-convex ReLU network is given by

C to NC:

$$W_{1i} = v_i^* / \sqrt{\|v_i^*\|}, \quad w_{2i} = \sqrt{\|v_i^*\|}$$
$$W_{1j} = w_i^* / \sqrt{\|w_i^*\|}, \quad w_{2j} = -\sqrt{\|w_i^*\|}.$$

- Optimal convex weights satisfy $v_i^* = \alpha_i q_i$ so that

$$\|v_i^*\|_2 = \alpha_i \|q_i\|_2 = \alpha_i \lambda.$$

Recall structure of **non-convex optima**:

$$\begin{aligned}\mathcal{O}_\lambda = \big\{ &(W_1, w_2) : \; f_{W_1, w_2}(X) = \hat{y}, \\ &\forall\, i \in \mathcal{S}_\lambda, W_{1i} = ({}^{\alpha_i}/\lambda)^{1/2} q_i, w_{2i} = (\alpha_i \lambda)^{1/2}, \alpha_i \geq 0 \\ &\forall\, i \in [2p] \setminus \mathcal{S}_\lambda, W_{1i} = 0, \; w_{2i} = 0 \big\}.\end{aligned}$$

Optimal Pruning

Optimal Pruning: the Polytope of Solutions

$$\mathcal{W}^*(\lambda) = \big\{\theta : \sum_{i=1}^{2p} D_i X \theta_i = \hat{y},$$
$$\forall\, i \in \mathcal{S}_\lambda, \theta_i = \alpha_i q_i, \alpha_i \geq 0,$$
$$\forall\, j \in [2p] \setminus \mathcal{S}_\lambda, \theta_j = 0\big\}$$

Optimal Pruning: the Polytope of Solutions

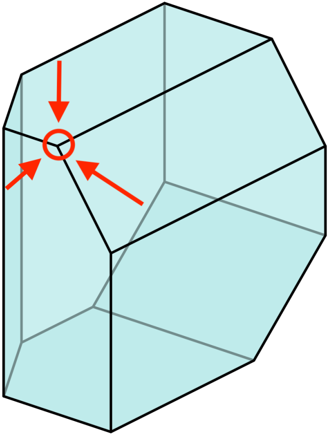
$$\mathcal{W}^*(\lambda) = \Big\{\theta : \sum_{i=1}^{2p} D_i X \theta_i = \hat{y},$$
$$\forall\, i \in \mathcal{S}_\lambda, \theta_i = \alpha_i q_i, \alpha_i \geq 0,$$
$$\forall\, j \in [2p] \setminus \mathcal{S}_\lambda, \theta_j = 0\Big\}$$

The C-ReLU optimal set is a convex polytope!

Optimal Pruning: the Polytope of Solutions

$$\mathcal{W}^*(\lambda) = \Big\{\theta : \sum_{i=1}^{2p} D_i X \theta_i = \hat{y},$$
$$\forall\, i \in \mathcal{S}_\lambda, \theta_i = \alpha_i q_i, \alpha_i \geq 0,$$
$$\forall\, j \in [2p] \setminus \mathcal{S}_\lambda, \theta_j = 0\Big\}$$

The C-ReLU optimal set is a convex polytope!



Optimal Pruning: Vertices

1. Stack the q_i vectors into a matrix $Q = \begin{bmatrix} | & & | \\ q_1 & \cdots & q_{2p} \\ | & & | \end{bmatrix}$.

Optimal Pruning: Vertices

1. Stack the q_i vectors into a matrix $Q = \begin{bmatrix} | & & | \\ q_1 & \cdots & q_{2p} \\ | & & | \end{bmatrix}$.
2. The C-ReLU Optimal Set in α space is then,

$$\begin{aligned} \mathcal{W}^*(\lambda) &= Q_{\mathcal{S}_\lambda} \{ \alpha \succeq 0 : \sum_{i \in \mathcal{S}_\lambda} (D_i X q_i) \alpha_i = \hat{y}, \} \\ &= Q_{\mathcal{S}_\lambda} \mathcal{P}_{\mathcal{S}_\lambda}. \end{aligned} \tag{2}$$

Optimal Pruning: Vertices

1. Stack the q_i vectors into a matrix $Q = \begin{bmatrix} | & & | \\ q_1 & \cdots & q_{2p} \\ | & & | \end{bmatrix}$.
2. The C-ReLU Optimal Set in α space is then,

$$\begin{aligned} \mathcal{W}^*(\lambda) &= Q_{\mathcal{S}_\lambda}\{\alpha \succeq 0 : \sum_{i \in \mathcal{S}_\lambda} (D_i X q_i)\alpha_i = \hat{y}, \} \\ &= Q_{\mathcal{S}_\lambda} \mathcal{P}_{\mathcal{S}_\lambda}. \end{aligned} \tag{2}$$

3. $\bar{\alpha} \in \mathcal{P}_{\mathcal{S}_\lambda}$ is a vertex iff $\{D_i X q_i\}_{\bar{\alpha}_i \neq 0}$ are linearly independent.

Optimal Pruning: Vertices

1. Stack the q_i vectors into a matrix $Q = \begin{bmatrix} | & & | \\ q_1 & \cdots & q_{2p} \\ | & & | \end{bmatrix}$.
2. The C-ReLU Optimal Set in α space is then,

$$\begin{aligned} \mathcal{W}^*(\lambda) &= Q_{\mathcal{S}_\lambda}\{\alpha \succeq 0 : \sum_{i \in \mathcal{S}_\lambda} (D_i X q_i)\alpha_i = \hat{y}, \} \\ &= Q_{\mathcal{S}_\lambda} \mathcal{P}_{\mathcal{S}_\lambda}. \end{aligned} \tag{2}$$

3. $\bar{\alpha} \in \mathcal{P}_{\mathcal{S}_\lambda}$ is a vertex iff $\{D_i X q_i\}_{\bar{\alpha}_i \neq 0}$ are linearly independent.

Are these vertices special in some way?

Optimal Pruning: Minimal Models

Definition: An optimal C-ReLU model θ^* is minimal if there does not exist another optimal model θ' with strictly smaller support.

Optimal Pruning: Minimal Models

Definition: An optimal C-ReLU model θ^* is minimal if there does not exist another optimal model θ' with strictly smaller support.

- NC-ReLU: minimal $\iff$ sparsest (neuron-wise) model.

Optimal Pruning: Minimal Models

Definition: An optimal C-ReLU model θ^* is minimal if there does not exist another optimal model θ' with strictly smaller support.

- NC-ReLU: minimal $\iff$ sparsest (neuron-wise) model.

Proposition 3.2 (informal): For $\lambda > 0$, $\theta \in \mathcal{W}^*(\lambda)$ is minimal iff the vectors $\{D_i X q_i\}_{\alpha_i \neq 0}$ are linearly independent.

Optimal Pruning: Minimal Models

Definition: An optimal C-ReLU model θ^* is minimal if there does not exist another optimal model θ' with strictly smaller support.

- NC-ReLU: minimal $\iff$ sparsest (neuron-wise) model.

Proposition 3.2 (informal): For $\lambda > 0$, $\theta \in \mathcal{W}^*(\lambda)$ is minimal iff the vectors $\{D_i X q_i\}_{\alpha_i \neq 0}$ are linearly independent.

- NC-ReLU: minimal if $(XW_{1i})_+$ are linearly independent.

Optimal Pruning: Minimal Models

Definition: An optimal C-ReLU model θ^* is minimal if there does not exist another optimal model θ' with strictly smaller support.

- NC-ReLU: minimal $\iff$ sparsest (neuron-wise) model.

Proposition 3.2 (informal): For $\lambda > 0$, $\theta \in \mathcal{W}^*(\lambda)$ is minimal iff the vectors $\{D_i X q_i\}_{\alpha_i \neq 0}$ are linearly independent.

- NC-ReLU: minimal if $(XW_{1i})_+$ are linearly independent.

Our Results:

1. We prove vertices of $\mathcal{W}^*(\lambda)$ are minimal models.

Optimal Pruning: Minimal Models

Definition: An optimal C-ReLU model θ^* is minimal if there does not exist another optimal model θ' with strictly smaller support.

- NC-ReLU: minimal $\iff$ sparsest (neuron-wise) model.

Proposition 3.2 (informal): For $\lambda > 0$, $\theta \in \mathcal{W}^*(\lambda)$ is minimal iff the vectors $\{D_i X q_i\}_{\alpha_i \neq 0}$ are linearly independent.

- NC-ReLU: minimal if $(XW_{1i})_+$ are linearly independent.

Our Results:

1. We prove vertices of $\mathcal{W}^*(\lambda)$ are minimal models.
2. There are at most n neurons in a minimal model.

Optimal Pruning: Minimal Models

Definition: An optimal C-ReLU model θ^* is minimal if there does not exist another optimal model θ' with strictly smaller support.

- NC-ReLU: minimal $\iff$ sparsest (neuron-wise) model.

Proposition 3.2 (informal): For $\lambda > 0$, $\theta \in \mathcal{W}^*(\lambda)$ is minimal iff the vectors $\{D_i X q_i\}_{\alpha_i \neq 0}$ are linearly independent.

- NC-ReLU: minimal if $(XW_{1i})_+$ are linearly independent.

Our Results:

1. We prove vertices of $\mathcal{W}^*(\lambda)$ are minimal models.
2. There are at most n neurons in a minimal model.
3. We give a poly-time algorithm for computing minimal models starting from any model θ.

Bonus: Optimal Pruning Pseudo-code

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$.
$\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**
  $i^k \leftarrow \arg\max_i \left\{ |\beta_i| : i \in \mathcal{A}_\lambda(\theta^k) \right\}$
  $t^k \leftarrow 1/|\beta_{i^k}|$
  $\theta^{k+1} \leftarrow \theta^k (1 - t^k \beta_i)$
  $k \leftarrow k + 1$
end while
Output: final weights θ^k

Bonus: Optimal Pruning Pseudo-code

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$.
$\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**
  $i^k \leftarrow \arg\max_i \left\{ |\beta_i| : i \in \mathcal{A}_\lambda(\theta^k) \right\}$
  $t^k \leftarrow 1/|\beta_{i^k}|$
  $\theta^{k+1} \leftarrow \theta^k (1 - t^k \beta_i)$
  $k \leftarrow k + 1$
end while
Output: final weights θ^k

Let $r = \text{rank}(X)$. Complexity to compute a minimal model:

$$O\left(d^3 r^3 (\frac{n}{r})^{3r} + (n^3 + nd) r (\frac{n}{r})^r \right).$$

Bonus: Complexity of Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$, $\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**
$\vdots$
end while
Output: final weights θ^k

- Computing $a_i = D_i X \theta_i^0$ for every neuron: $O(ndp)$

Bonus: Complexity of Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.

$k \leftarrow 0$, $\theta^k \leftarrow \theta$.

while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**

$\quad\vdots$

end while

Output: final weights θ^k

- Computing $a_i = D_i X \theta_i^0$ for every neuron: $O(ndp)$
- Checking for linear dependence: at most $2p$ times, do

Bonus: Complexity of Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$, $\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**

$\vdots$

end while
Output: final weights θ^k

- Computing $a_i = D_i X \theta_i^0$ for every neuron: $O(ndp)$
- Checking for linear dependence: at most $2p$ times, do
    - check (at most) $n + 1$ a_i vectors for linearly dependence.

Bonus: Complexity of Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$, $\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**
$\quad\vdots$
end while
Output: final weights θ^k

- Computing $a_i = D_i X \theta_i^0$ for every neuron: $O(ndp)$
- Checking for linear dependence: at most $2p$ times, do
    - check (at most) $n + 1$ a_i vectors for linearly dependence.
    - Form matrix A and take SVD to compute null space: $O(n^3)$.

Bonus: Complexity of Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$, $\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**
$\quad \vdots$
end while
Output: final weights θ^k

- Computing $a_i = D_i X \theta_i^0$ for every neuron: $O(ndp)$
- Checking for linear dependence: at most $2p$ times, do
  - check (at most) $n + 1$ a_i vectors for linearly dependence.
  - Form matrix A and take SVD to compute null space: $O(n^3)$.
  - Prune neuron: update at most n weights.

Bonus: Complexity of Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$, $\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**
  $\vdots$
end while
Output: final weights θ^k

- Computing $a_i = D_i X \theta_i^0$ for every neuron: $O(ndp)$
- Checking for linear dependence: at most $2p$ times, do
  - check (at most) $n + 1$ a_i vectors for linearly dependence.
  - Form matrix A and take SVD to compute null space: $O(n^3)$.
  - Prune neuron: update at most n weights.

Total complexity: $O(ndp + n^3 p)$.

Bonus: Sub-Optimal Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.

$k \leftarrow 0$.

$\theta^k \leftarrow \theta$.

while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**

  $i^k \leftarrow \arg\max_i \left\{ |\beta_i| : i \in \mathcal{A}_\lambda(\theta^k) \right\}$

  $t^k \leftarrow 1/|\beta_{i^k}|$

  $\theta^{k+1} \leftarrow \theta^k (1 - t^k \beta_i)$

  $k \leftarrow k + 1$

end while

Output: final weights θ^k

Bonus: Sub-Optimal Pruning

Algorithm Pruning solutions

Input: data matrix X, solution θ.
$k \leftarrow 0$.
$\theta^k \leftarrow \theta$.
while $\exists \beta \neq 0$ s.t. $\sum_{i \in \mathcal{A}_\lambda(\theta^k)} \beta_i D_i X \theta_i^k = 0$ **do**
  $i^k \leftarrow \arg\max_i \left\{ |\beta_i| : i \in \mathcal{A}_\lambda(\theta^k) \right\}$
  $t^k \leftarrow 1/|\beta_{i^k}|$
  $\theta^{k+1} \leftarrow \theta^k (1 - t^k \beta_i)$
  $k \leftarrow k + 1$
end while
Output: final weights θ^k

Approximate with least-squares fit:

$$\hat{\beta} \in \arg\min_{\beta} \frac{1}{2} \| \sum_{i \in \mathcal{A}_\lambda(\theta^k) \setminus j} \beta_i D_i X \theta_i^k - D_j X \theta_j \|_2^2$$

Bonus: Sub-optimal Pruning

Approximate with least-squares fit:

$$\hat{\beta} \in \arg\min_{\beta} \frac{1}{2} \| \sum_{i \in \mathcal{A}_\lambda(\theta^k) \setminus j} \beta_i D_i X \theta_i^k - D_j X \theta_j \|_2^2$$

- Algorithm is just structured pruning with a correction step!

Bonus: Sub-optimal Pruning

Approximate with least-squares fit:

$$\hat{\beta} \in \arg\min_{\beta} \frac{1}{2} \| \sum_{i \in \mathcal{A}_\lambda(\theta^k) \setminus j} \beta_i D_i X \theta_i^k - D_j X \theta_j \|_2^2$$

- Algorithm is just structured pruning with a correction step!
- We use existing literature for structured pruning to select j.

Bonus: Sub-optimal Pruning

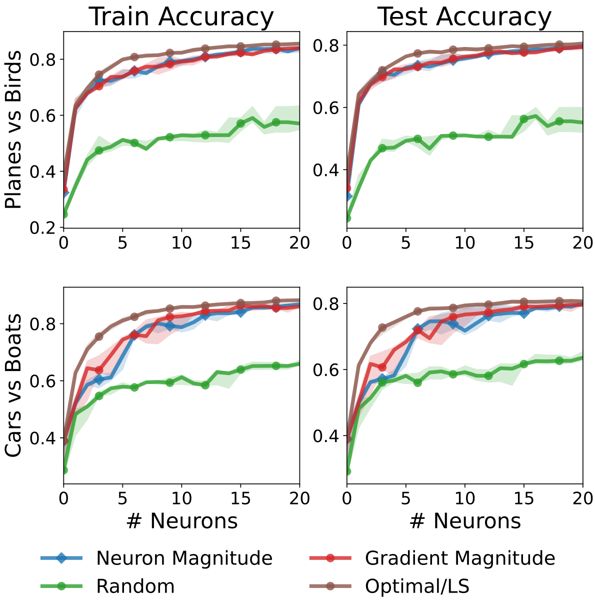
Approximate with least-squares fit:

$$\hat{\beta} \in \arg\min_{\beta} \frac{1}{2} \| \sum_{i \in \mathcal{A}_\lambda(\theta^k) \backslash j} \beta_i D_i X \theta_i^k - D_j X \theta_j \|_2^2$$

- Algorithm is just structured pruning with a correction step!
- We use existing literature for structured pruning to select j.
- Brute-force search works best:

$$\arg\min_{j} \left\{ \min_{\beta} \frac{1}{2} \| \sum_{i \in \mathcal{A}_\lambda(\theta^k) \backslash j} \beta_i D_i X \theta_i^k - D_j X \theta_j \|_2^2 \right\}$$

Neuron Pruning: Performance on CIFAR-10



Convex Reformulations of Deep ReLU Networks

Deep Reformulations: Setup

- Let $X^{(l)}$ and $T^{(l)}$ be tensors of order $l+1$ indexed by $i_0, \ldots, i_l$.
- We assume $R^{(l)}_{i_1,\ldots,i_{l-1}} \in \mathbb{R}^{n \times d_0}$ and $T^{(l)}_{i_1,\ldots,i_{l-1}} \in \mathbb{R}^{d_0 \times d_l}$.

We equip these tensors with the reduction product

$$R^{(l)} \odot T^{(l)} = \sum_{i_1,\ldots,i_{l-1}} R^{(l)}_{i_1,\ldots,i_{l-1}} T^{(l)}_{i_1,\ldots,i_{l-1}},$$

- This sums over the product of all the matrix slices $R^{(l)}_{i_1,\ldots,i_{l-1}}$ and $T^{(l)}_{i_1,\ldots,i_{l-1}}$.

Deep Reformulations: Recursive Patterns

- Let $\mathcal{D}_X^{(1)}$ be the set of achievable ReLU patterns in the first layer.
- Let $X^{(1)} = X \in \mathbb{R}^{n \times d_0}$.
- Define $X_{i_1,\ldots,i_l}^{(l+1)} = D_{i_l}^{(l)} X_{i_1,\ldots,i_{l-1}}^{(l)}$ so that we have,

$$X_{i_1}^{(2)} = D_{i_1}^{(1)} X^{(1)} = D_{i_1}^{(1)} X$$
$$X_{i_1,i_2}^{(3)} = D_{i_2}^{(2)} X_{i_1}^{(2)} = D_{i_2}^{(2)} D_{i_1}^{(1)} X.$$

- Here, $D_{i_l}^{(l)} \in \mathcal{D}_X^{(l)}$ is the set of ReLU patterns achievable by our tensor product in the l^{th} layer,

$$\mathcal{D}_X^{(l)} = \left\{ \mathbb{1}\left(X^{(l)} \odot T^{(l)} \geq 0 \right) : T^{(l)} \in \mathbb{R}^{d_0 \times \cdots \times d_l} \right\}.$$

Deep Reformulations: Tensor Decomposition

Each tensor $T_{i_l}^{(l)}$ is contained in at least one activation cone,

$$\mathcal{K}_{i_l}^{(l)} = \left\{ T_{i_l}^{(l)} \in \mathbb{R}^{d_0 \times \cdots \times d_{(l-1)}} : (2D_{i_l}^{(l)} - I) \left[X^{(l)} \odot T_{i_l}^{(l)} \right] \geq 0 \right\}.$$

Now, let $\mathcal{G}^{(1)} = \mathbb{R}^{d_0 \times d_1}$ and define

$$\begin{aligned} \mathcal{G}^{(l+1)} := \Bigg\{ & T^{(l+1)} \in \mathbb{R}^{d_0 \times p_1 \ldots \times p_l \times d_{l+1}} \\ & : \exists\, T^{(l)} \in \mathcal{G}^{(l)} \text{ where } \mathcal{I}_{j_l}^{(l)} = \left\{ i_l : T_{i_l}^{(l)} \in \mathcal{K}_{j_l}^{(l)} \right\}, \\ & \exists\, W^{(l+1)} \in \mathbb{R}^{d_l \times d_{l+1}}, \\ & \text{s.t. } T_{j_1, \ldots, j_l}^{(l+1)} = \sum_{i_l \in \mathcal{I}_{j_l}^{(l)}} T_{j_1, \ldots, j_{l-1}, i_l}^{(l)} \otimes W_{i_l}^{(l+1)}, \\ & \text{and } \sum_{j_l = 1}^{p_l} \left| \mathcal{I}_{j_l}^{(l)} \right| \leq d_l \Bigg\}. \end{aligned}$$

Deep Reformulations: Layer-Merging Lemma

Lemma (Rank-Controlled Layer Elimination)

Let $T^{(l)} \in \mathcal{G}^{(l)}$. *Then the activations at layer* $l+2$ *are given by*

$$Z^{(l+2)} = \left(\sum_{i_l=1}^{d_l} \left(X^{(l)} \odot T_{i_l}^{(l)} \right)_+ W_{i_l}^{(l+1)} \right)_+, \tag{3}$$

if and only if the activations are also equal to

$$Z^{(l+2)} = \left(X^{(l+1)} \odot T^{(l+1)} \right)_+,$$

for some $T^{(l+1)} \in \mathcal{G}^{(l+1)}$.