Phase Transitions for Feature Learning in Neural Networks

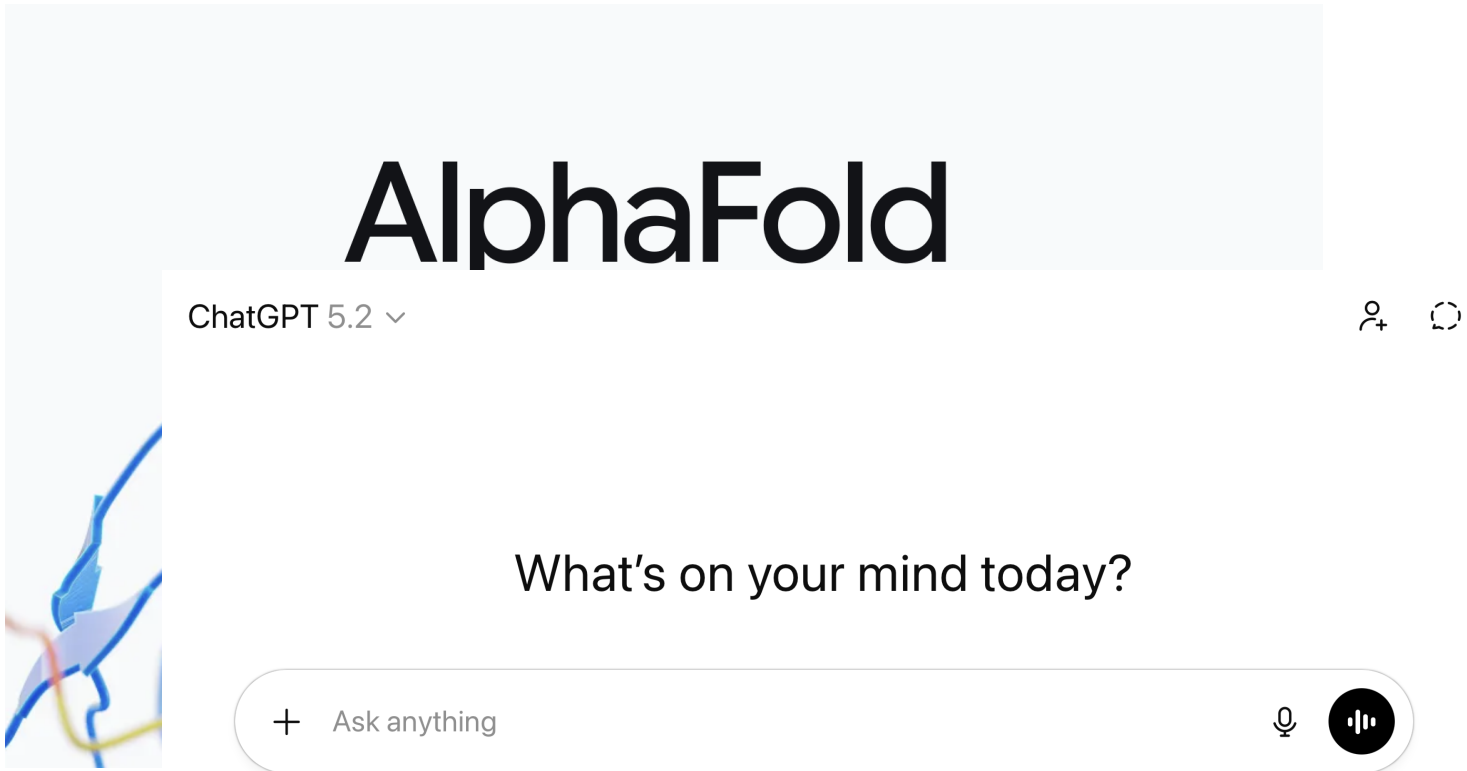
Zihao Wang (Stanford University)

Center for Computational Mathematics, Flatiron Institute
March 2026

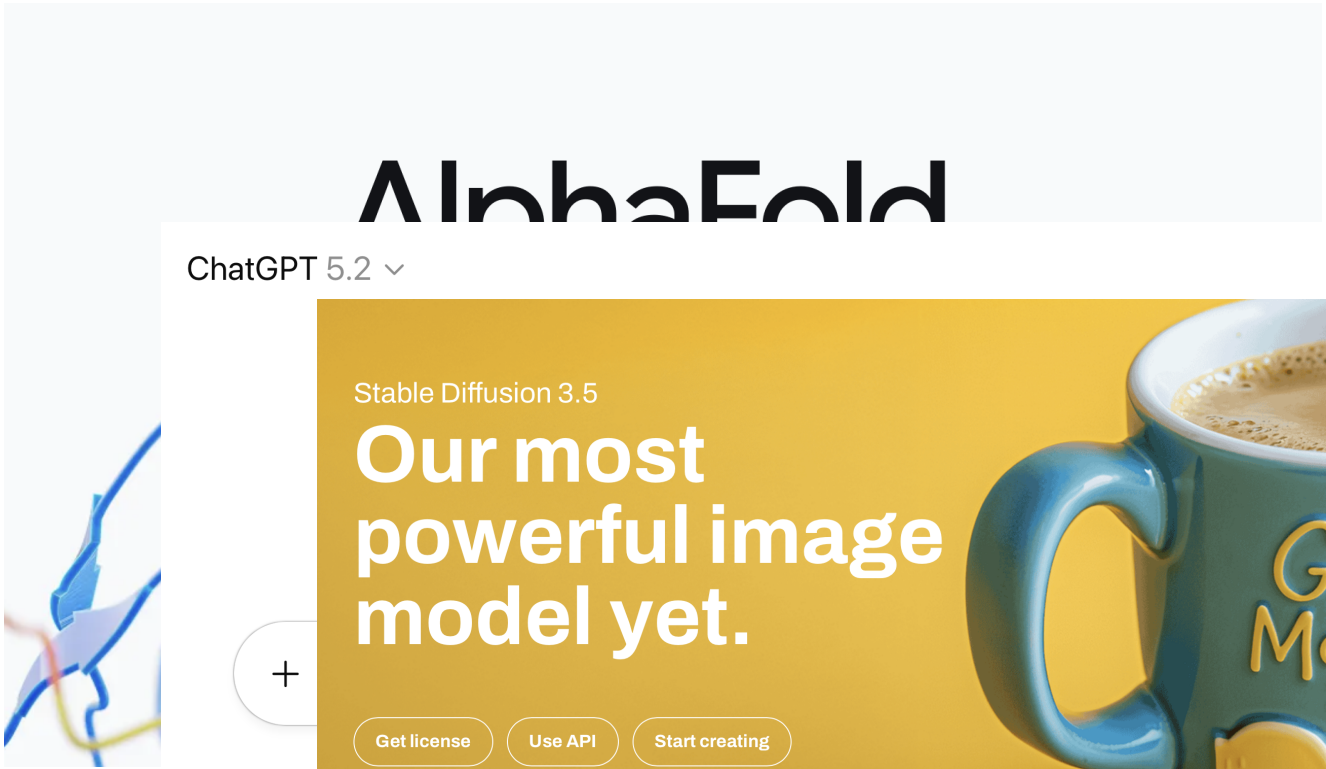
Joint work with Andrea Montanari
(Stanford University)

Empirical success

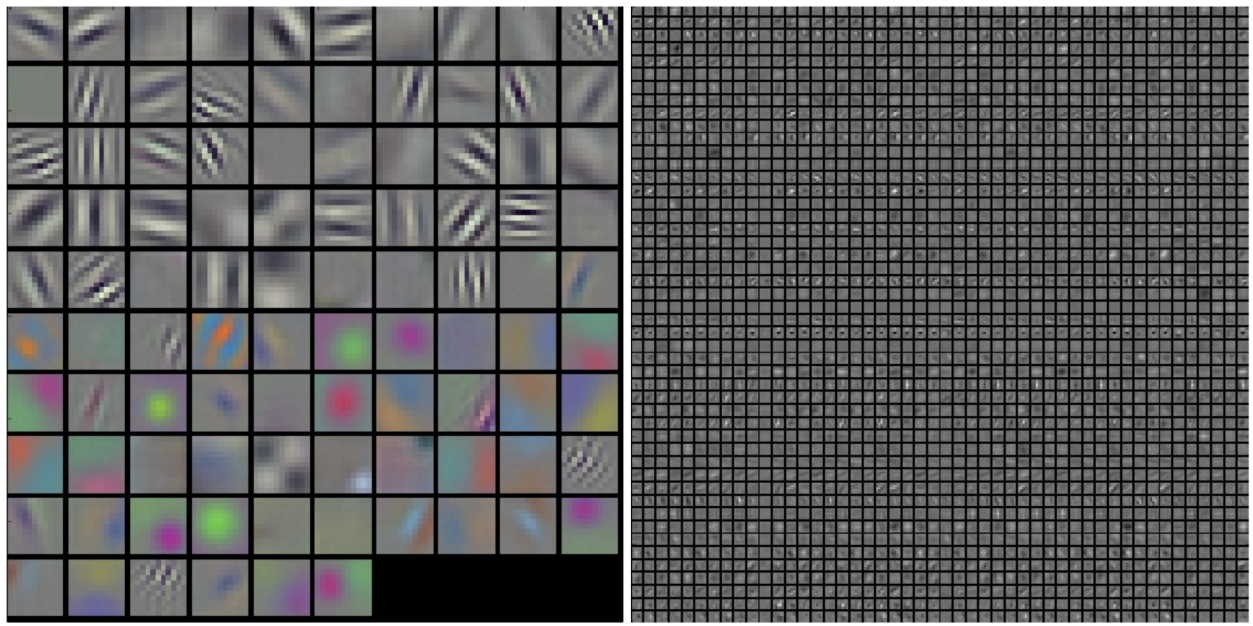
Empirical success



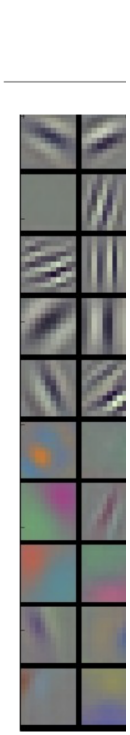
Empirical success

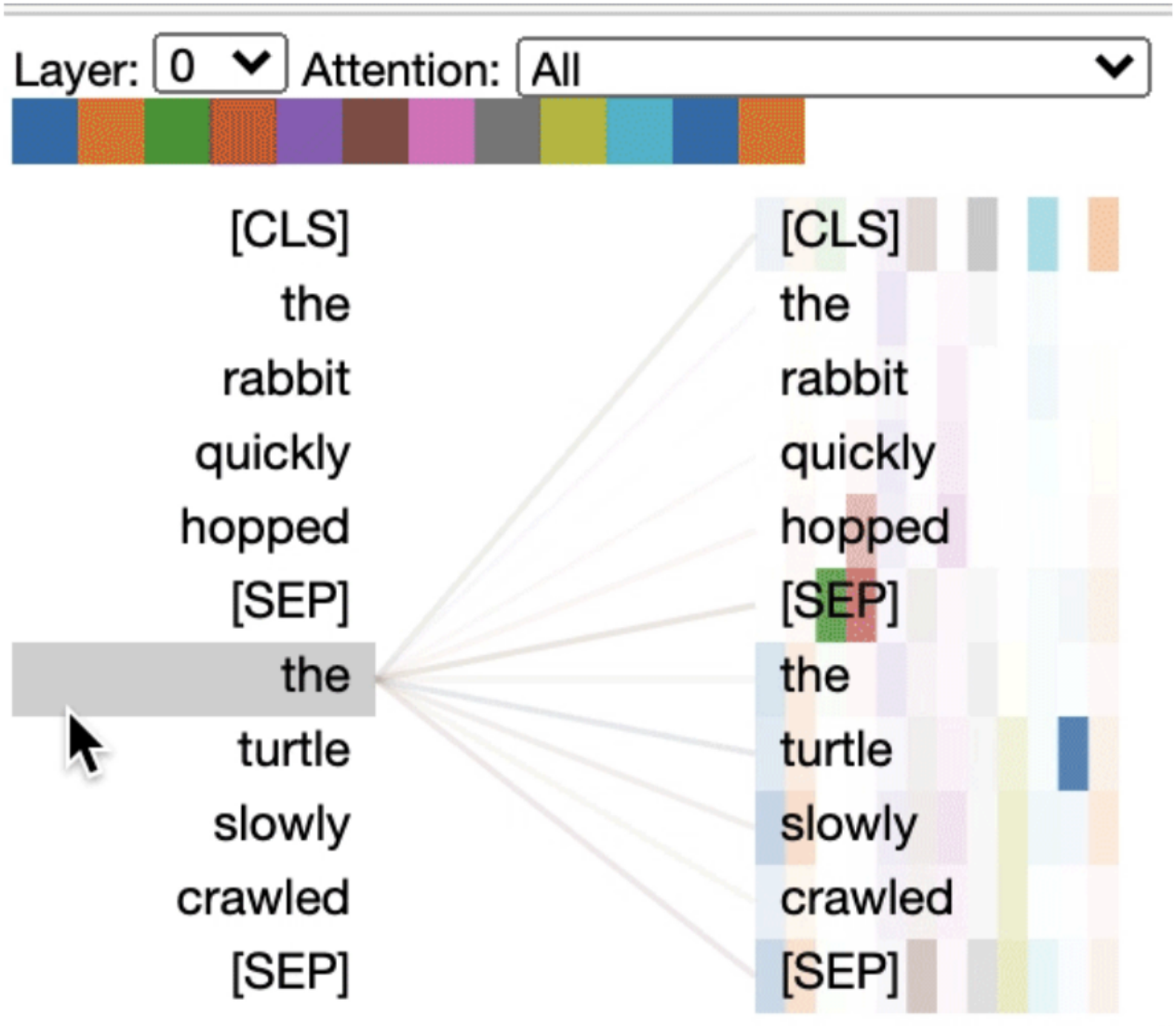


... from learning representations of data

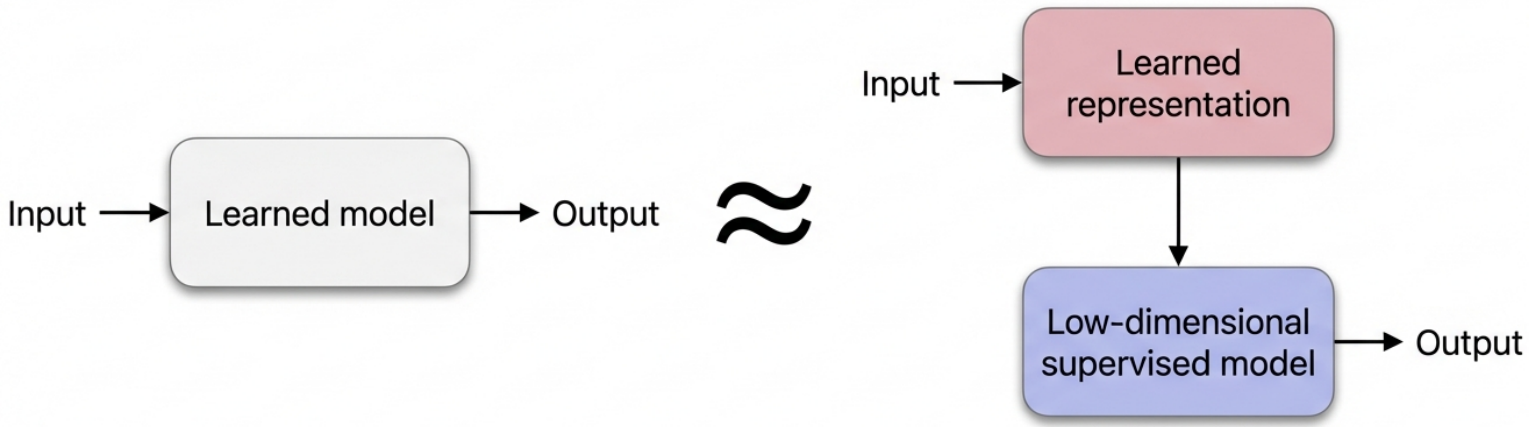


... from learning representations of data



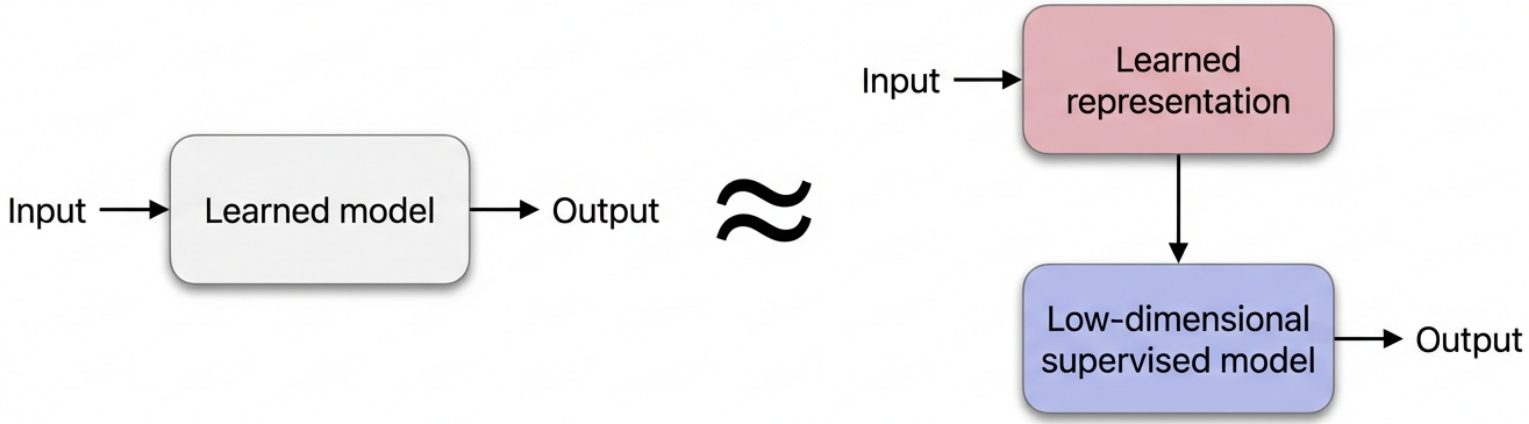


Folklore



First learn representations, then fit a low-dimensional model over those representations!

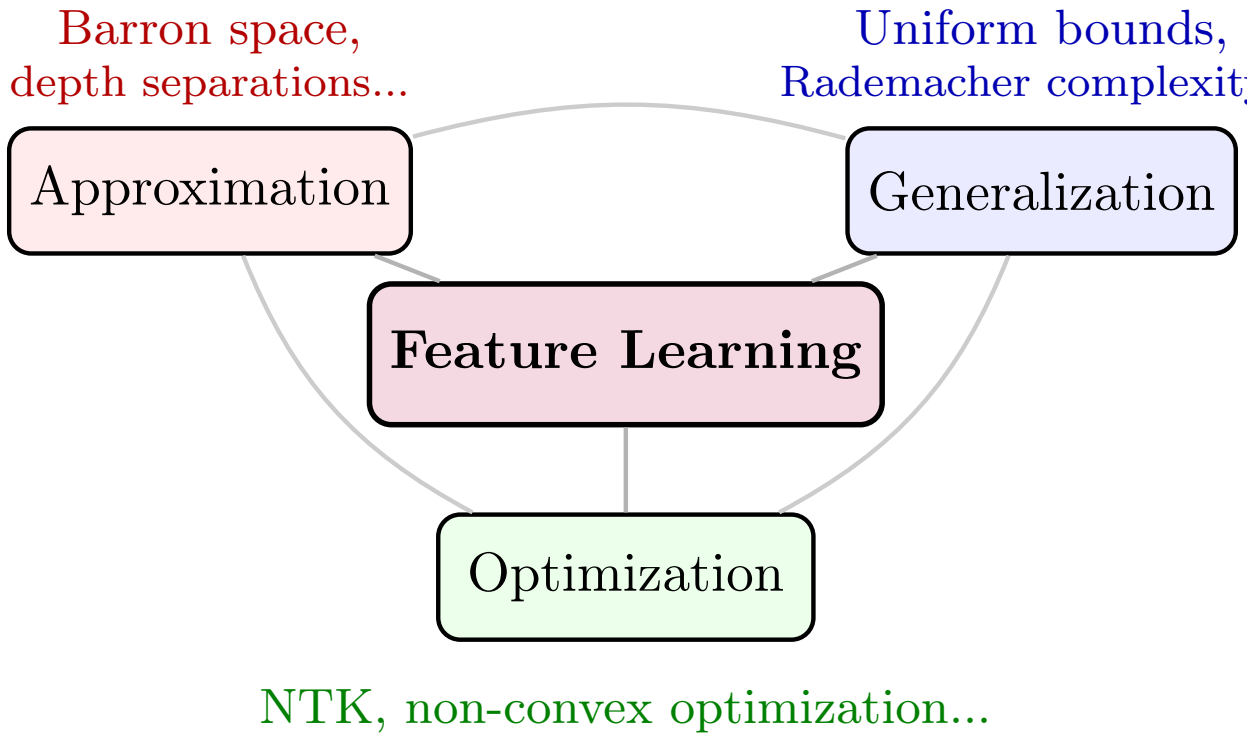
Folklore



First learn representations, then fit a low-dimensional model over those representations!

Feature learning

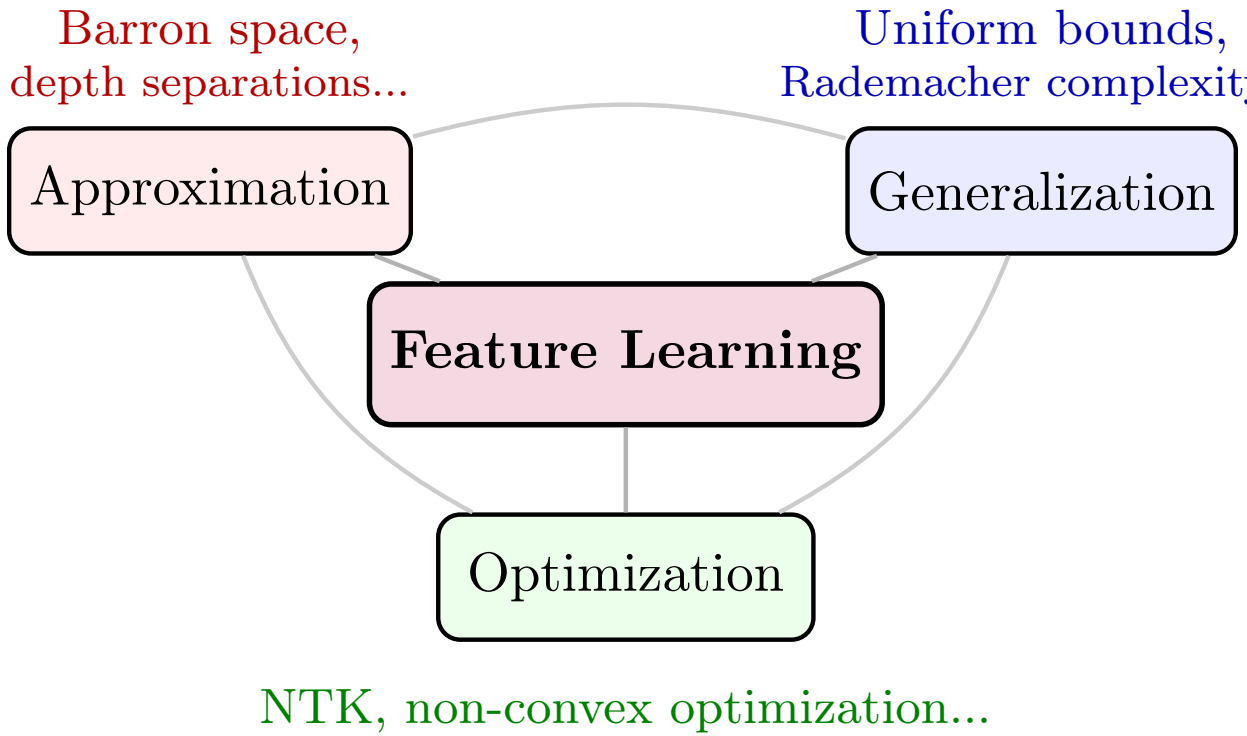
- Feature learning: Learning those representations from data.



In the feature learning regime, approximation, generalization, and optimization are **entangled** with each other.

Feature learning

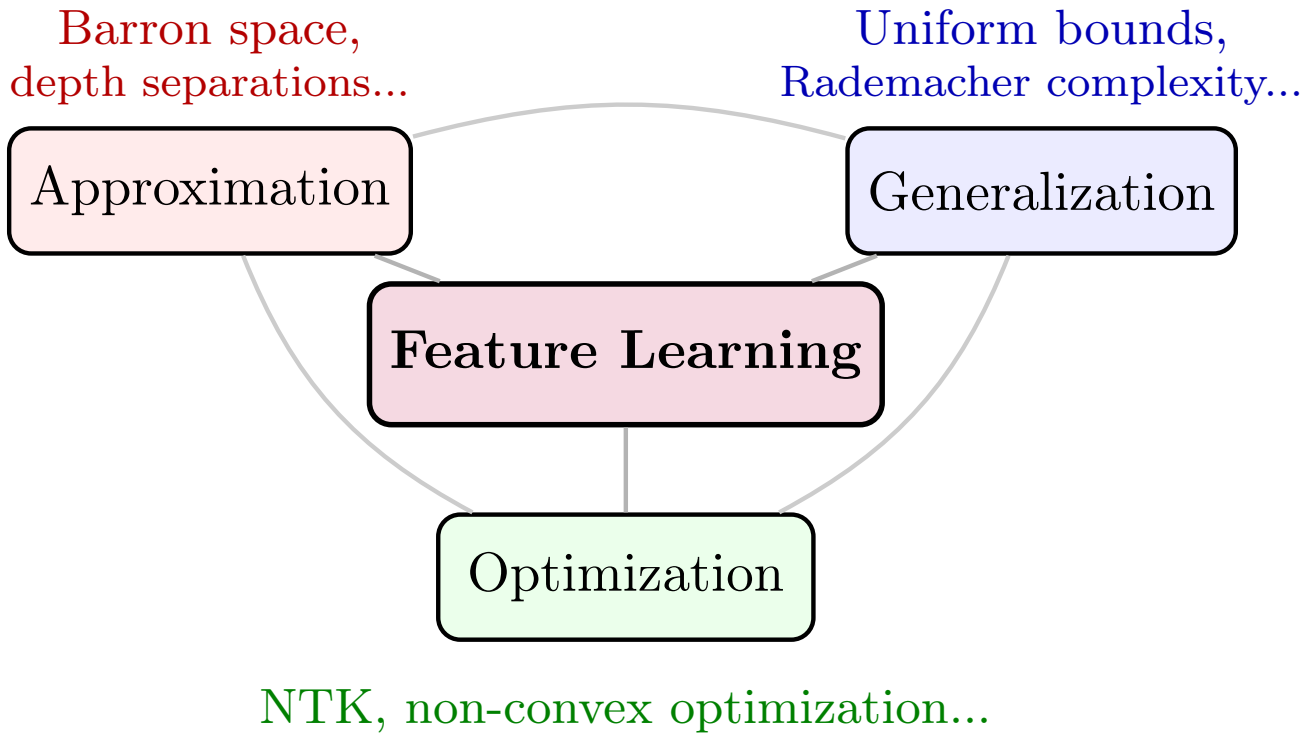
- Feature learning: Learning those representations from data.



In the feature learning regime, approximation, generalization, and optimization are **entangled** with each other.

Feature learning

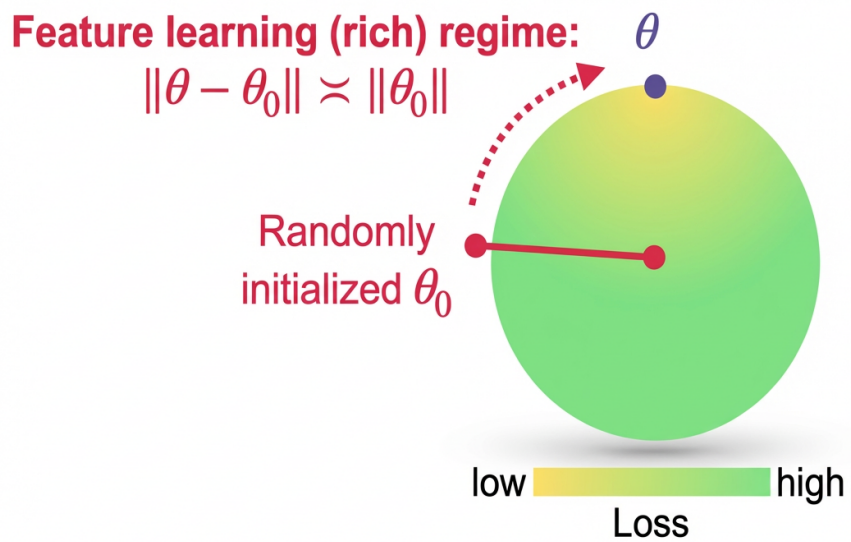
- Feature learning: Learning those representations from data.



In the feature learning regime, approximation, generalization, and optimization are **entangled** with each other.

Feature learning: not NTK/RF kernel regime

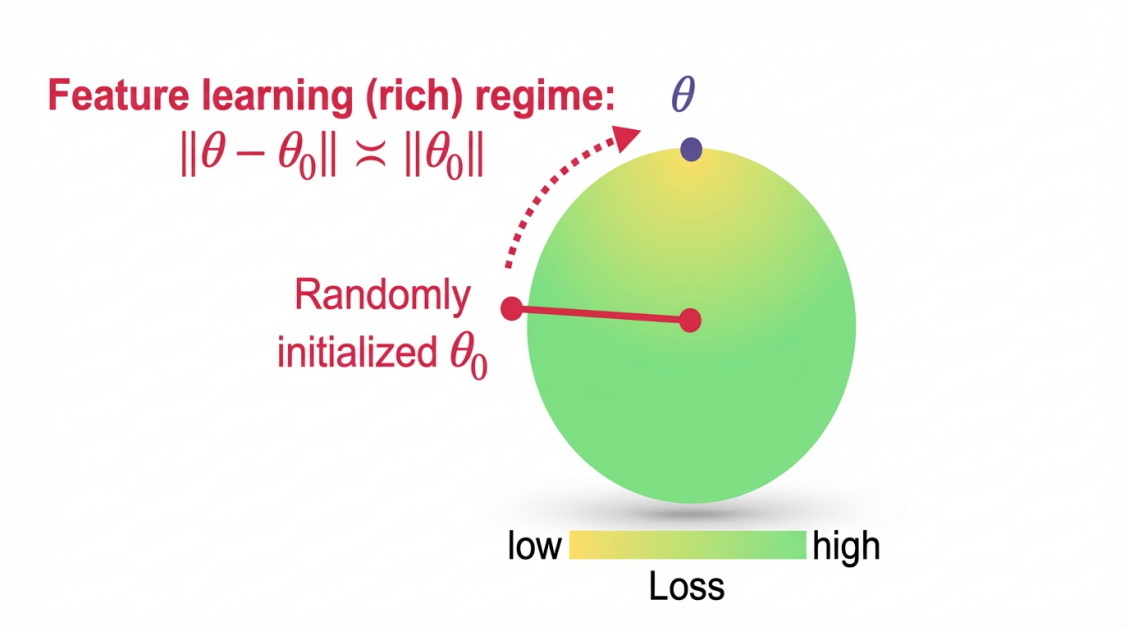
- Consider a neural network $\mathsf{f} : \mathcal{X} \times \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$ that takes input $\mathbf{x} \in \mathcal{X}$ and a parametrization $\boldsymbol{\theta} \in \mathbb{R}^{\mathsf{D}}$ and outputs a scalar.



- Kernel/NTK/Lazy training regime: $\|\theta - \theta_0\| \ll \|\theta_0\|$.

Feature learning: not NTK/RF kernel regime

- Consider a neural network $\mathsf{f} : \mathcal{X} \times \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$ that takes input $\mathbf{x} \in \mathcal{X}$ and a parametrization $\boldsymbol{\theta} \in \mathbb{R}^{\mathsf{D}}$ and outputs a scalar.



- Kernel/NTK/Lazy training regime: $\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\| \ll \|\boldsymbol{\theta}_0\|$.

Research questions

We aim to answer the following questions:

- Formalize 'feature learning'; a proper mathematical model.
- (Optimization) How models learn features through gradient descent? Understand the **loss landscape and training dynamics**.
- (Generalization) The required **sample size** to learn features. How does this scale with target task, loss function, activation, etc.?

Research questions

We aim to answer the following questions:

- Formalize 'feature learning'; a proper mathematical model.
- (Optimization) How models learn features through gradient descent? Understand the **loss landscape and training dynamics**.
- (Generalization) The required **sample size** to learn features. How does this scale with target task, loss function, activation, etc.?

Research questions

We aim to answer the following questions:

- Formalize 'feature learning'; a proper mathematical model.
- (Optimization) How models learn features through gradient descent? Understand the **loss landscape and training dynamics**.
- (Generalization) The required **sample size** to learn features. How does this scale with target task, loss function, activation, etc.?

Outline

Multi-index models

Consider multi-index models as our targets for feature learning.

Setup: Observe n i.i.d. samples $(x_i, y_i)_{i \le n}$:

- Covariates: $x_i \sim \mathsf{N}(0, I_d)$ (high-dimensional, no structure)
- Responses depend on a low-dimensional projection:

$$y_i = h(\Theta_*^\top x_i, \varepsilon_i), \quad \varepsilon_i \sim \mathsf{N}(0,1)$$

- $\Theta_* \in \mathbb{R}^{d \times k}$ $(k \ll d)$
- $h : \mathbb{R}^{k+1} \to \mathbb{R}$ is the link function.

Multi-index models

Consider multi-index models as our targets for feature learning.

Setup: Observe n i.i.d. samples $(x_i, y_i)_{i \le n}$:

- Covariates: $x_i \sim \mathsf{N}(0, I_d)$ (high-dimensional, no structure)
- Responses depend on a low-dimensional projection:

$$y_i = h(\Theta_*^\top x_i, \varepsilon_i), \quad \varepsilon_i \sim \mathsf{N}(0,1)$$

- $\Theta_* \in \mathbb{R}^{d \times k}$ $(k \ll d)$
- $h : \mathbb{R}^{k+1} \to \mathbb{R}$ is the link function.

Multi-index models

Consider multi-index models as our targets for feature learning.

Setup: Observe n i.i.d. samples $(x_i, y_i)_{i \le n}$:

- Covariates: $x_i \sim \mathsf{N}(0, I_d)$ (high-dimensional, no structure)
- Responses depend on a low-dimensional projection:

$$y_i = h(\Theta_*^\top x_i, \varepsilon_i), \quad \varepsilon_i \sim \mathsf{N}(0,1)$$

- $\Theta_* \in \mathbb{R}^{d \times k}$ $(k \ll d)$
- $h : \mathbb{R}^{k+1} \to \mathbb{R}$ is the link function.

Multi-index models

Consider multi-index models as our targets for feature learning.

Setup: Observe n i.i.d. samples $(x_i, y_i)_{i \le n}$:

- Covariates: $x_i \sim N(0, I_d)$ (high-dimensional, no structure)
- Responses depend on a low-dimensional projection:

$$y_i = h\big(\Theta_*^\top x_i,\, \varepsilon_i\big), \quad \varepsilon_i \sim N(0,1)$$

- $\Theta_* \in \mathbb{R}^{d \times k}$ $(k \ll d)$
- $h : \mathbb{R}^{k+1} \to \mathbb{R}$ is the link function.

Multi-index models

Consider multi-index models as our targets for feature learning.

Setup: Observe n i.i.d. samples $(x_i, y_i)_{i \le n}$:

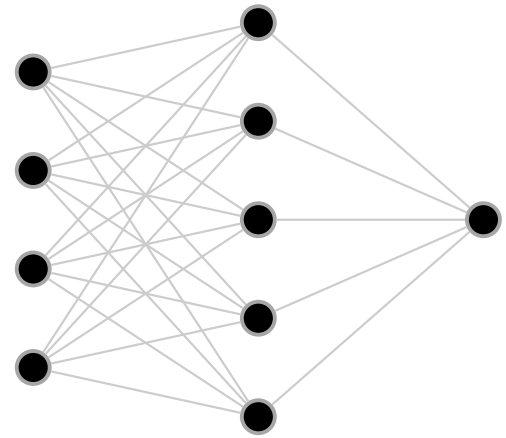
- Covariates: $x_i \sim \mathsf{N}(0, I_d)$ (high-dimensional, no structure)
- Responses depend on a low-dimensional projection:

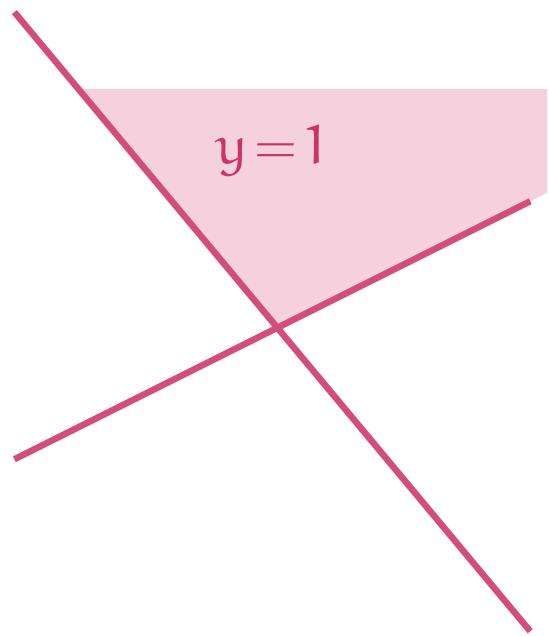
$$y_i = h(\Theta_*^\top x_i, \varepsilon_i), \quad \varepsilon_i \sim \mathsf{N}(0,1)$$

- $\Theta_* \in \mathbb{R}^{d \times k}$ $(k \ll d)$
- $h : \mathbb{R}^{k+1} \to \mathbb{R}$ is the link function.

Examples of multi-index models

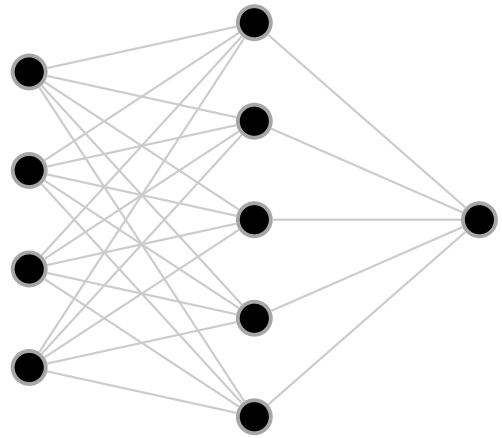
- **Single-index** $(k = 1)$:
  (e.g., phase retrieval) $y = (\theta_*^\top x)^2$
- **Neural networks** $(O(1)$ neurons$)$:
  $y = \sum_{j=1}^k a_j \sigma(\theta_{*j}^\top x) + \varepsilon$
- **Intersection of halfspaces**:
  $y = \prod_{j=1}^k 1_{\{\theta_{*j}^\top x > 0\}}$
- **Polynomials** on $O(1)$ directions:
  $y = p(\Theta_*^\top x)$

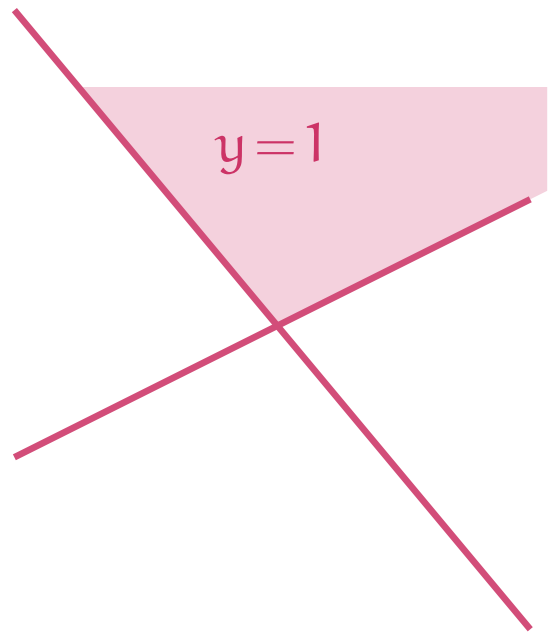




Examples of multi-index models

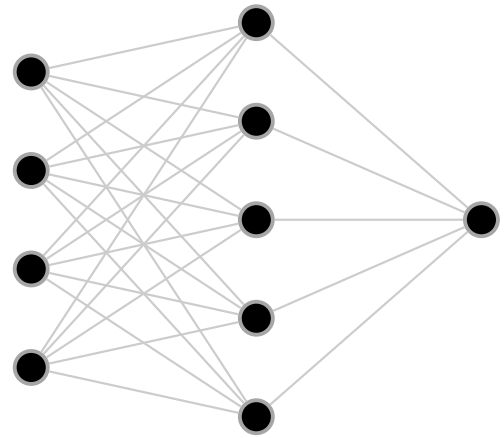
- **Single-index** ($k = 1$):
  (e.g., phase retrieval) $y = (\theta_*^\top x)^2$
- **Neural networks** ($O(1)$ neurons):
  $y = \sum_{j=1}^{k} a_j \sigma(\theta_{*j}^\top x) + \varepsilon$
- **Intersection of halfspaces**:
  $y = \prod_{j=1}^{k} 1_{\{\theta_{*j}^\top x > 0\}}$
- **Polynomials** on $O(1)$ directions:
  $y = p(\Theta_*^\top x)$

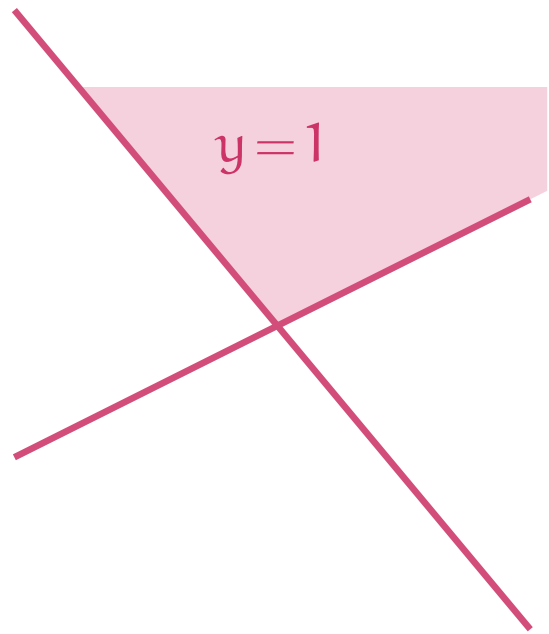




Examples of multi-index models

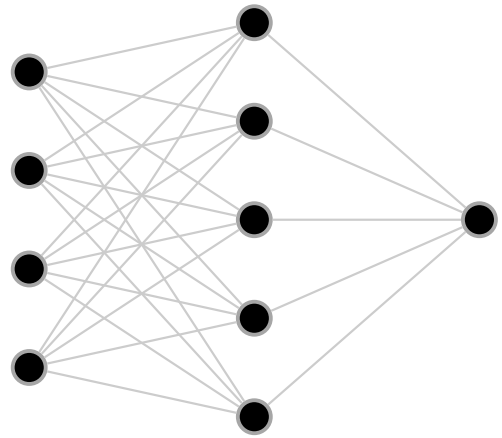
- **Single-index** $(k = 1)$:
  (e.g., phase retrieval) $y = (\theta_*^\top x)^2$
- **Neural networks** $(O(1)$ neurons$)$:
  $y = \sum_{j=1}^{k} a_j \sigma(\theta_{*j}^\top x) + \varepsilon$
- **Intersection of halfspaces**:
  $y = \prod_{j=1}^{k} 1_{\{\theta_{*j}^\top x > 0\}}$
- **Polynomials** on $O(1)$ directions:
  $y = p(\Theta_*^\top x)$

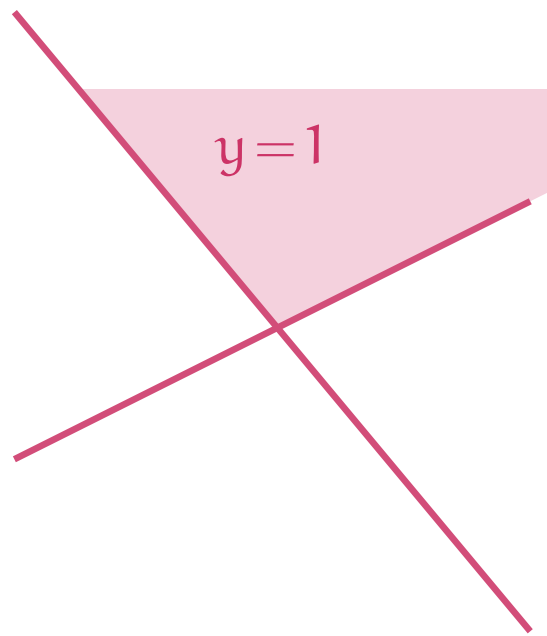




Examples of multi-index models

- **Single-index** $(k = 1)$:
  (e.g., phase retrieval) $y = (\theta_*^\top x)^2$
- **Neural networks** $(O(1)$ neurons$)$:
  $y = \sum_{j=1}^{k} a_j \sigma(\theta_{*j}^\top x) + \varepsilon$
- **Intersection of halfspaces**:
  $y = \prod_{j=1}^{k} \mathbf{1}_{\{\theta_{*j}^\top x > 0\}}$
- **Polynomials** on $O(1)$ directions:
  $y = p(\Theta_*^\top x)$





The learning problem

Feature learning = Learning the latent span($\boldsymbol{\Theta}_*$)

Question: How many samples do we need to perform feature learning?

- Simple enough to be tractable mathematically
- Complex enough that NTK/RF/KRR cannot learn[1]

Information-theoretic answer: $n = \Theta(d)$ (counting degrees of freedom) [Barbier, Krzakala, Macris+ '19], [Aubin, Maillard, Barbier+ '19]

[1] [Yehudai-Shamir '19], [Ghorbani-Mei-Misiakiewicz-Montanari '19]

The learning problem

Feature learning = Learning the latent span($\boldsymbol{\Theta}_*$)

Question: How many samples do we need to perform feature learning?

- Simple enough to be tractable mathematically
- Complex enough that NTK/RF/KRR cannot learn[1]

Information-theoretic answer: $n = \Theta(d)$ (counting degrees of freedom) [Barbier, Krzakala, Macris+ '19], [Aubin, Maillard, Barbier+ '19]

[1] [Yehudai-Shamir '19], [Ghorbani-Mei-Misiakiewicz-Montanari '19]

The learning problem

Feature learning = Learning the latent span($\boldsymbol{\Theta}_*$)

Question: How many samples do we need to perform feature learning?

- Simple enough to be tractable mathematically
- Complex enough that NTK/RF/KRR cannot learn[1]

Information-theoretic answer: $n = \Theta(d)$ (counting degrees of freedom) [Barbier, Krzakala, Macris+ '19], [Aubin, Maillard, Barbier+ '19]

[1] [Yehudai-Shamir '19], [Ghorbani-Mei-Misiakiewicz-Montanari '19]

The learning problem

Feature learning = Learning the latent span($\boldsymbol{\Theta}_*$)

Question: How many samples do we need to perform feature learning?

- Simple enough to be tractable mathematically
- Complex enough that NTK/RF/KRR cannot learn[1]

Information-theoretic answer: $n = \Theta(d)$ (counting degrees of freedom) [Barbier, Krzakala, Macris+ '19], [Aubin, Maillard, Barbier+ '19]

[1] [Yehudai-Shamir '19], [Ghorbani-Mei-Misiakiewicz-Montanari '19]

Prior work: Learning multi-index models

Insight: There exist sharp **phase transitions** in $\delta = n/d$.

- **Statistical threshold δ_{IT}:**
  Below δ_{IT}, learning is information-theoretically impossible.

- **Algorithmic threshold δ_{alg}:**
  Below δ_{alg}, no polynomial-time algorithm can learn Θ_*.

Gap: Often $\delta_{\mathsf{IT}} < \delta_{\mathsf{alg}}$ — computational hardness

[Mondelli, Montanari '18], [Lu, Li '19], [Chen, Meka '20], [Damian, Pillaud-Vivien, Lee, Bruna '24], [Damian, Lee, Bruna '25], [Troiani, Dandi+ '25], [Kovačević, Zhang, Mondelli '25], [Defilippis, Dandi+ '25], ...

Neural Networks? (we want an analogous δ_{NN})

Prior work: Learning multi-index models

Insight: There exist sharp **phase transitions** in $\delta = n/d$.

- **Statistical threshold δ_{IT}:**
  Below δ_{IT}, learning is information-theoretically impossible.

- **Algorithmic threshold δ_{alg}:**
  Below δ_{alg}, no polynomial-time algorithm can learn Θ_*.

Gap: Often $\delta_{IT} < \delta_{alg}$ — computational hardness

[Mondelli, Montanari '18], [Lu, Li '19], [Chen, Meka '20], [Damian, Pillaud-Vivien, Lee, Bruna '24], [Damian, Lee, Bruna '25], [Troiani, Dandi+ '25], [Kovačević, Zhang, Mondelli '25], [Defilippis, Dandi+ '25], ...

Neural Networks? (we want an analogous δ_{NN})

Prior work: Learning multi-index models

Insight: There exist sharp **phase transitions** in $\delta = n/d$.

- **Statistical threshold δ_{IT}:**
  Below δ_{IT}, learning is information-theoretically impossible.
- **Algorithmic threshold δ_{alg}:**
  Below δ_{alg}, no polynomial-time algorithm can learn Θ_*.

Gap: Often $\delta_{\mathsf{IT}} < \delta_{\mathsf{alg}}$ — computational hardness

[Mondelli, Montanari '18], [Lu, Li '19], [Chen, Meka '20], [Damian, Pillaud-Vivien, Lee, Bruna '24], [Damian, Lee, Bruna '25], [Troiani, Dandi+ '25], [Kovačević, Zhang, Mondelli '25], [Defilippis, Dandi+ '25], ...

Neural Networks? (we want an analogous δ_{NN})

Prior work: Learning multi-index models

Insight: There exist sharp **phase transitions** in $\delta = n/d$.

- **Statistical threshold δ_{IT}:**
  Below δ_{IT}, learning is information-theoretically impossible.
- **Algorithmic threshold δ_{alg}:**
  Below δ_{alg}, no polynomial-time algorithm can learn $\boldsymbol{\Theta}_*$.

Gap: Often $\delta_{\mathsf{IT}} < \delta_{\mathsf{alg}}$ — computational hardness

[Mondelli, Montanari '18], [Lu, Li '19], [Chen, Meka '20], [Damian, Pillaud-Vivien, Lee, Bruna '24], [Damian, Lee, Bruna '25], [Troiani, Dandi+ '25], [Kovačević, Zhang, Mondelli '25], [Defilippis, Dandi+ '25], ...

Neural Networks? (we want an analogous δ_{NN})

Prior work: Learning multi-index models

Insight: There exist sharp **phase transitions** in $\delta = n/d$.

- **Statistical threshold δ_{IT}:**
  Below δ_{IT}, learning is information-theoretically impossible.
- **Algorithmic threshold δ_{alg}:**
  Below δ_{alg}, no polynomial-time algorithm can learn $\boldsymbol{\Theta}_*$.

Gap: Often $\delta_{\mathsf{IT}} < \delta_{\mathsf{alg}}$ — computational hardness

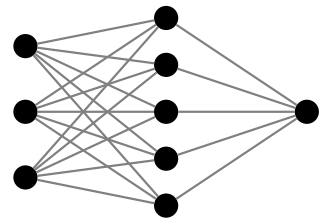
[Mondelli, Montanari '18], [Lu, Li '19], [Chen, Meka '20], [Damian, Pillaud-Vivien, Lee, Bruna '24], [Damian, Lee, Bruna '25], [Troiani, Dandi+ '25], [Kovačević, Zhang, Mondelli '25], [Defilippis, Dandi+ '25], ...

Neural Networks? (we want an analogous δ_{NN})

Neural network setup

Two-layer neural network:

$$f_\Theta(x) = \frac{1}{m} \sum_{j=1}^{m} a_j \sigma(\theta_j^\top x + b_j)$$



- $\Theta = [\theta_1, \ldots, \theta_m] \in \mathbb{R}^{d \times m}$: first-layer weights
- σ: activation function (e.g., GeLU, ReLU, ...)
- (a_j, b_j): fixed second-layer weights and biases

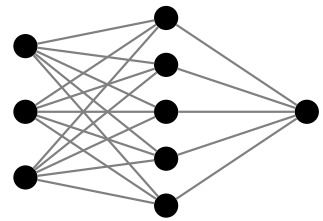
Training: Only train Θ via gradient descent on empirical risk

$$\text{Risk}(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\Theta(x_i))$$

Neural network setup

Two-layer neural network:

$$f_\Theta(x) = \frac{1}{m} \sum_{j=1}^{m} a_j \sigma(\theta_j^\top x + b_j)$$



- $\Theta = [\theta_1, \ldots, \theta_m] \in \mathbb{R}^{d \times m}$: first-layer weights
- σ: activation function (e.g., GeLU, ReLU, ...)
- (a_j, b_j): fixed second-layer weights and biases

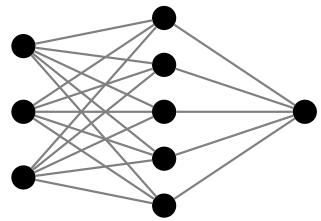
Training: Only train Θ via gradient descent on empirical risk

$$\text{Risk}(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\Theta(x_i))$$

Neural network setup

Two-layer neural network:

$$f_{\Theta}(x) = \frac{1}{m} \sum_{j=1}^{m} a_j \sigma(\theta_j^\top x + b_j)$$



- $\Theta = [\theta_1, \ldots, \theta_m] \in \mathbb{R}^{d \times m}$: first-layer weights
- σ: activation function (e.g., GeLU, ReLU, ...)
- (a_j, b_j): fixed second-layer weights and biases

Training: Only train Θ via gradient descent on empirical risk

$$\text{Risk}(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_{\Theta}(x_i))$$

Training protocol

Full-batch gradient descent:

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{\Theta}(t) - \eta \nabla_{\boldsymbol{\Theta}} \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

Initialization: $\theta_j \overset{\text{i.i.d.}}{\sim} \mathsf{Unif}(\mathbb{S}^{d-1})$

Proportional asymptotics:

$$n, d \to \infty, \quad n/d \to \delta \in (0, \infty)$$

Training protocol

Full-batch gradient descent:

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{\Theta}(t) - \eta \nabla_{\boldsymbol{\Theta}} \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

Initialization: $\theta_j \overset{\text{i.i.d.}}{\sim} \mathsf{Unif}(\mathbb{S}^{d-1})$

Proportional asymptotics:

$$n, d \to \infty, \quad n/d \to \delta \in (0, \infty)$$

Training protocol

Full-batch gradient descent:

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{\Theta}(t) - \eta \nabla_{\boldsymbol{\Theta}} \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

Initialization: $\theta_j \overset{\text{i.i.d.}}{\sim} \mathsf{Unif}(\mathbb{S}^{d-1})$

Proportional asymptotics:

$$n, d \to \infty, \quad n/d \to \delta \in (0, \infty)$$

Prior work: Feature learning in neural networks

Extensive literature: [Ben Arous+ '21], [Damian+ '22], [Ba+ '22], [Abbe+ '23], [Bietti+ '23], [**Wang**+ '23], [Berthier+ '24], [Dandi+ '24], [Lee+ '24], [Fu, **Wang**+ '24], [Montanari, Urbani '25], ...

Understanding are limited; results are far from optimal.

Previous: [Zhang*, **Wang***, Fu, Lee '25]

- For generic[2] multi-index models:

$$n = \widetilde{O}(d) \text{ samples suffice for GD on NNs}$$

- Sharp in **leading order** for generic cases, but loose in constants.

[2] Generative leap exponent ≤ 2 with no generative-staircase structure; covers almost all common examples.

Prior work: Feature learning in neural networks

Extensive literature: [Ben Arous+ '21], [Damian+ '22], [Ba+ '22], [Abbe+ '23], [Bietti+ '23], [**Wang**+ '23], [Berthier+ '24], [Dandi+ '24], [Lee+ '24], [Fu, **Wang**+ '24], [Montanari, Urbani '25], ...

Understanding are limited; results are far from optimal.

Previous: [Zhang*, **Wang***, Fu, Lee '25]

- For generic[2] multi-index models:

$$n = \widetilde{O}(d) \text{ samples suffice for GD on NNs}$$

- Sharp in **leading order** for generic cases, but loose in constants.

[2] Generative leap exponent ≤ 2 with no generative-staircase structure; covers almost all common examples.

Prior work: Feature learning in neural networks

Extensive literature: [Ben Arous+ '21], [Damian+ '22], [Ba+ '22], [Abbe+ '23], [Bietti+ '23], [**Wang**+ '23], [Berthier+ '24], [Dandi+ '24], [Lee+ '24], [Fu, **Wang**+ '24], [Montanari, Urbani '25], ...

Understanding are limited; results are far from optimal.

Previous: [Zhang*, **Wang***, Fu, Lee '25]

- For generic[2] multi-index models:

$$n = \widetilde{O}(d) \text{ samples suffice for GD on NNs}$$

- Sharp in **leading order** for generic cases, but loose in constants.

[2] Generative leap exponent ≤ 2 with no generative-staircase structure; covers almost all common examples.

Neural networks?

Central question: What is the learning threshold δ_{NN} for gradient descent on neural networks?

- Do NNs achieve the same threshold as δ_{alg}?
- If not, what is the gap? How does it depend on:
  - Architecture (activation, width, ...)
  - Algorithm (loss, initialization, stepsize, ...)
- What is the mechanism that determines δ_{NN}?

Neural networks?

Central question: What is the learning threshold δ_{NN} for gradient descent on neural networks?

- Do NNs achieve the same threshold as δ_{alg}?
- If not, what is the gap? How does it depend on:
  - Architecture (activation, width, ...)
  - Algorithm (loss, initialization, stepsize, ...)
- What is the mechanism that determines δ_{NN}?

Neural networks?

Central question: What is the learning threshold δ_{NN} for gradient descent on neural networks?

- Do NNs achieve the same threshold as δ_{alg}?
- If not, what is the gap? How does it depend on:
  - Architecture (activation, width, ...)
  - Algorithm (loss, initialization, stepsize, ...)
- What is the mechanism that determines δ_{NN}?

Neural networks?

Central question: What is the learning threshold δ_{NN} for gradient descent on neural networks?

- Do NNs achieve the same threshold as δ_{alg}?
- If not, what is the gap? How does it depend on:
  - Architecture (activation, width, ...)
  - Algorithm (loss, initialization, stepsize, ...)
- What is the mechanism that determines δ_{NN}?

Our results (informal)

Main contribution: Assuming the width $\mathfrak{m}$ is one or a large enough constant, given the knowledge of

- Target function $\mathsf{h}(\cdot)$,
- Loss function $\ell(\cdot, \cdot)$,
- Activation function $\sigma(\cdot)$,
- Learning rate η

We derive an **exact, explicit formula** for computing δ_{NN}:

- $\delta > \delta_{\mathsf{NN}} \Rightarrow$ Weak recovery[3] of $\boldsymbol{\Theta}_*$
- $\delta < \delta_{\mathsf{NN}} \Rightarrow$ Hardness of weak recovery of $\boldsymbol{\Theta}_*$

[3] Non-vanishing correlation between $\boldsymbol{\Theta}_*$ and the learned weights $\boldsymbol{\Theta}$.

Our results (informal)

Main contribution: Assuming the width $\mathfrak{m}$ is one or a large enough constant, given the knowledge of

- Target function $\mathsf{h}(\cdot)$,
- Loss function $\ell(\cdot, \cdot)$,
- Activation function $\sigma(\cdot)$,
- Learning rate η

We derive an **exact, explicit formula** for computing δ_{NN}:

- $\delta > \delta_{\mathsf{NN}} \Rightarrow$ Weak recovery[3] of $\boldsymbol{\Theta}_*$
- $\delta < \delta_{\mathsf{NN}} \Rightarrow$ Hardness of weak recovery of $\boldsymbol{\Theta}_*$

[3] Non-vanishing correlation between $\boldsymbol{\Theta}_*$ and the learned weights $\boldsymbol{\Theta}$.

Our results (informal)

Main contribution: Assuming the width $\mathfrak{m}$ is one or a large enough constant, given the knowledge of

- Target function $\mathfrak{h}(\cdot)$,
- Loss function $\ell(\cdot, \cdot)$,
- Activation function $\sigma(\cdot)$,
- Learning rate η

We derive an **exact, explicit formula** for computing δ_{NN}:

- $\delta > \delta_{\mathsf{NN}} \Rightarrow$ Weak recovery[3] of $\boldsymbol{\Theta}_*$
- $\delta < \delta_{\mathsf{NN}} \Rightarrow$ Hardness of weak recovery of $\boldsymbol{\Theta}_*$

[3] Non-vanishing correlation between $\boldsymbol{\Theta}_*$ and the learned weights $\boldsymbol{\Theta}$.

Our results (informal)

Main contribution: Assuming the width $\mathfrak{m}$ is one or a large enough constant, given the knowledge of

- Target function $\mathfrak{h}(\cdot)$,
- Loss function $\ell(\cdot, \cdot)$,
- Activation function $\sigma(\cdot)$,
- Learning rate η

We derive an **exact, explicit formula** for computing δ_{NN}:

- $\delta > \delta_{\mathsf{NN}} \Rightarrow$ Weak recovery[3] of $\boldsymbol{\Theta}_*$
- $\delta < \delta_{\mathsf{NN}} \Rightarrow$ Hardness of weak recovery of $\boldsymbol{\Theta}_*$

[3] Non-vanishing correlation between $\boldsymbol{\Theta}_*$ and the learned weights $\boldsymbol{\Theta}$.

Our results (informal)

Main contribution: Assuming the width $\mathfrak{m}$ is one or a large enough constant, given the knowledge of

- Target function $\mathfrak{h}(\cdot)$,
- Loss function $\ell(\cdot, \cdot)$,
- Activation function $\sigma(\cdot)$,
- Learning rate η

We derive an **exact, explicit formula** for computing δ_{NN}:

- $\delta > \delta_{\mathsf{NN}} \Rightarrow$ Weak recovery[3] of $\boldsymbol{\Theta}_*$
- $\delta < \delta_{\mathsf{NN}} \Rightarrow$ Hardness of weak recovery of $\boldsymbol{\Theta}_*$

[3] Non-vanishing correlation between $\boldsymbol{\Theta}_*$ and the learned weights $\boldsymbol{\Theta}$.

Our results (informal)

Main contribution: Assuming the width $\mathfrak{m}$ is one or a large enough constant, given the knowledge of

- Target function $\mathfrak{h}(\cdot)$,
- Loss function $\ell(\cdot, \cdot)$,
- Activation function $\sigma(\cdot)$,
- Learning rate η

We derive an **exact, explicit formula** for computing δ_{NN}:

- $\delta > \delta_{\mathsf{NN}} \Rightarrow$ Weak recovery[3] of Θ_*
- $\delta < \delta_{\mathsf{NN}} \Rightarrow$ Hardness of weak recovery of Θ_*

[3] Non-vanishing correlation between Θ_* and the learned weights Θ.

Our results (informal)

Main contribution: Assuming the width $\mathfrak{m}$ is one or a large enough constant, given the knowledge of

- Target function $\mathsf{h}(\cdot)$,
- Loss function $\ell(\cdot, \cdot)$,
- Activation function $\sigma(\cdot)$,
- Learning rate η

We derive an **exact, explicit formula** for computing δ_{NN}:

- $\delta > \delta_{\mathsf{NN}} \Rightarrow$ Weak recovery[3] of $\boldsymbol{\Theta}_*$
- $\delta < \delta_{\mathsf{NN}} \Rightarrow$ Hardness of weak recovery of $\boldsymbol{\Theta}_*$

[3] Non-vanishing correlation between $\boldsymbol{\Theta}_*$ and the learned weights $\boldsymbol{\Theta}$.

Properties of δ_{NN}

1. **Computable:**
   δ_{NN} can be efficiently computed from the problem setup.

2. **Spectral interpretation:**
   δ_{NN} = some BBP threshold[4] for the Hessian of empirical risk.

3. **Sub-optimal:** $\delta_{\mathsf{alg}} < \delta_{\mathsf{NN}}$
   The gap depends on activation, loss, initialization, ...

[4] Baik-Ben Arous-Péché

Properties of δ_{NN}

1. **Computable:**
   δ_{NN} can be efficiently computed from the problem setup.

2. **Spectral interpretation:**
   δ_{NN} = some BBP threshold[4] for the Hessian of empirical risk.

3. **Sub-optimal:** $\delta_{\mathsf{alg}} < \delta_{\mathsf{NN}}$
   The gap depends on activation, loss, initialization, ...

[4] Baik-Ben Arous-Péché

Properties of δ_{NN}

1. **Computable:**
   δ_{NN} can be efficiently computed from the problem setup.

2. **Spectral interpretation:**
   δ_{NN} = some BBP threshold[4] for the Hessian of empirical risk.

3. **Sub-optimal:** $\delta_{\mathsf{alg}} < \delta_{\mathsf{NN}}$
   The gap depends on activation, loss, initialization, ...

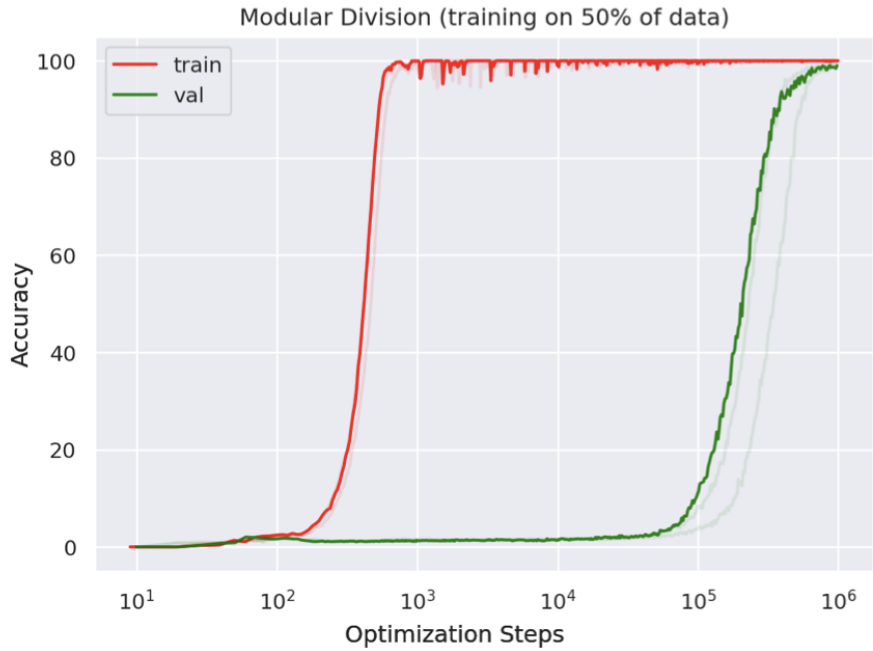
[4] Baik-Ben Arous-Péché

Grokking phenomenon

Grokking: Delayed generalization

- Training error drops quickly
- Test error stays high for long time
- Suddenly: test error drops

Example: Modular division task, trained on transformer architecture.



- Red: train accuracy
- Green: validation accuracy
- Gap of $\sim 10^3\times$ in steps

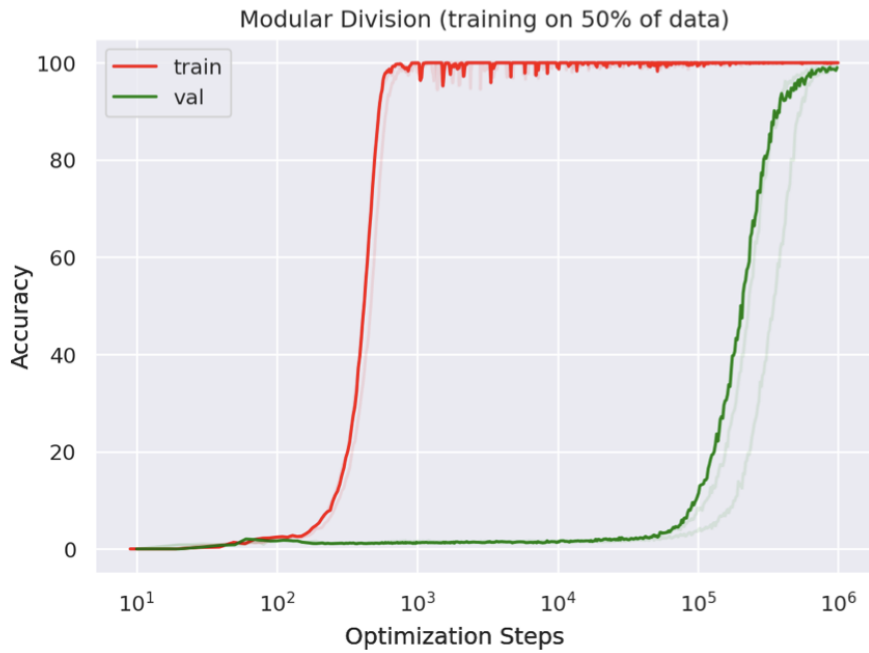
[Power, Burda, Edwards+ '22], [Liu, Kitouni, Nolte+ '22], ...

Grokking phenomenon

Grokking: Delayed generalization

- Training error drops quickly
- Test error stays high for long time
- Suddenly: test error drops

Example: Modular division task, trained on transformer architecture.



- Red: train accuracy
- Green: validation accuracy
- Gap of $\sim 10^3\times$ in steps

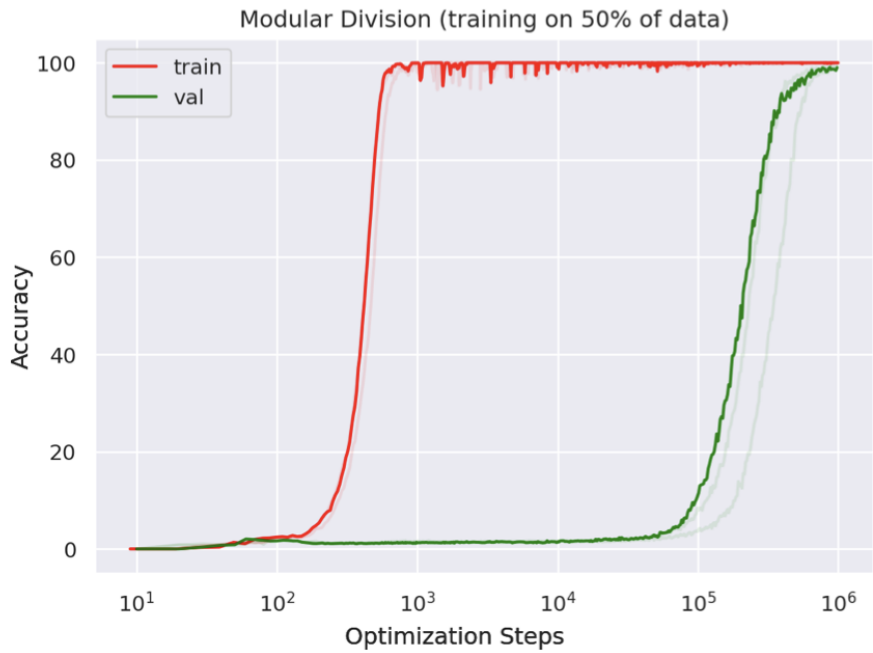
[Power, Burda, Edwards+ '22], [Liu, Kitouni, Nolte+ '22], ...

Grokking phenomenon

Grokking: Delayed generalization

- Training error drops quickly
- Test error stays high for long time
- Suddenly: test error drops

Example: Modular division task, trained on transformer architecture.



- Red: train accuracy
- Green: validation accuracy
- Gap of $\sim 10^3\times$ in steps

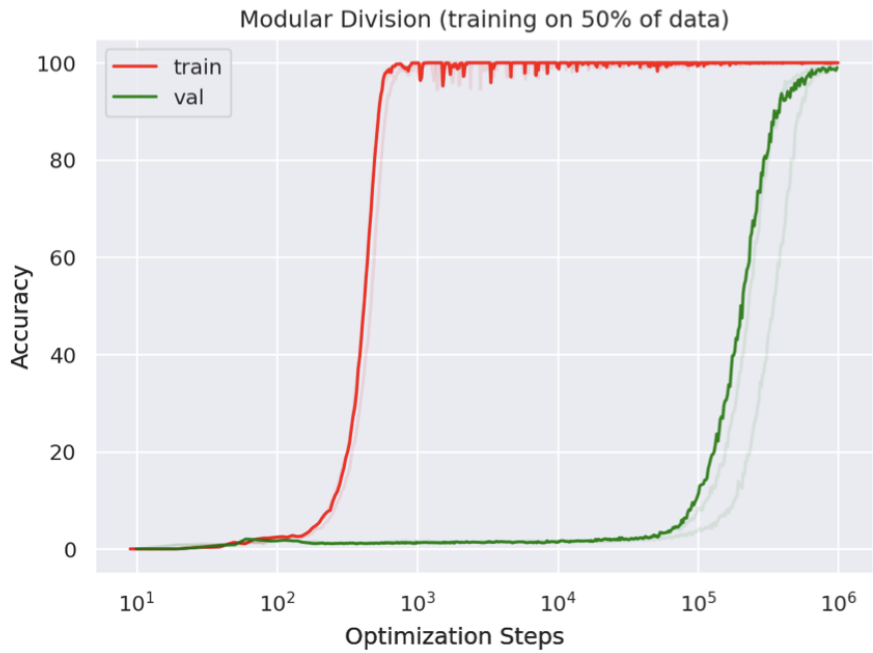
[Power, Burda, Edwards+ '22], [Liu, Kitouni, Nolte+ '22], ...

Grokking phenomenon

Grokking: Delayed generalization

- Training error drops quickly
- Test error stays high for long time
- Suddenly: test error drops

Example: Modular division task, trained on transformer architecture.



- Red: train accuracy
- Green: validation accuracy
- Gap of $\sim 10^3\times$ in steps

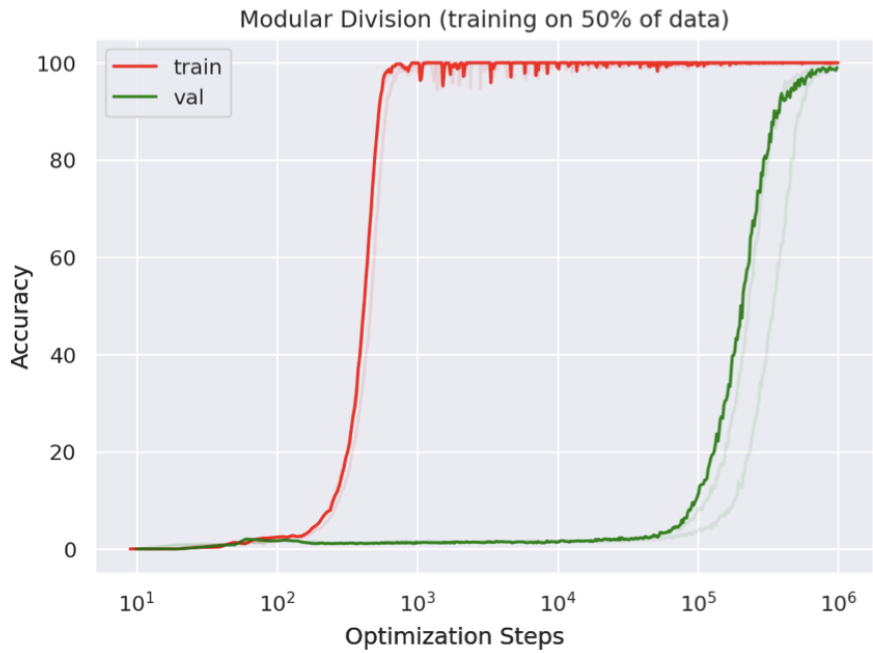
[Power, Burda, Edwards+ '22], [Liu, Kitouni, Nolte+ '22], ...

Grokking phenomenon

Grokking: Delayed generalization

- Training error drops quickly
- Test error stays high for long time
- Suddenly: test error drops

Example: Modular division task, trained on transformer architecture.



- Red: train accuracy
- Green: validation accuracy
- Gap of $\sim 10^3\times$ in steps

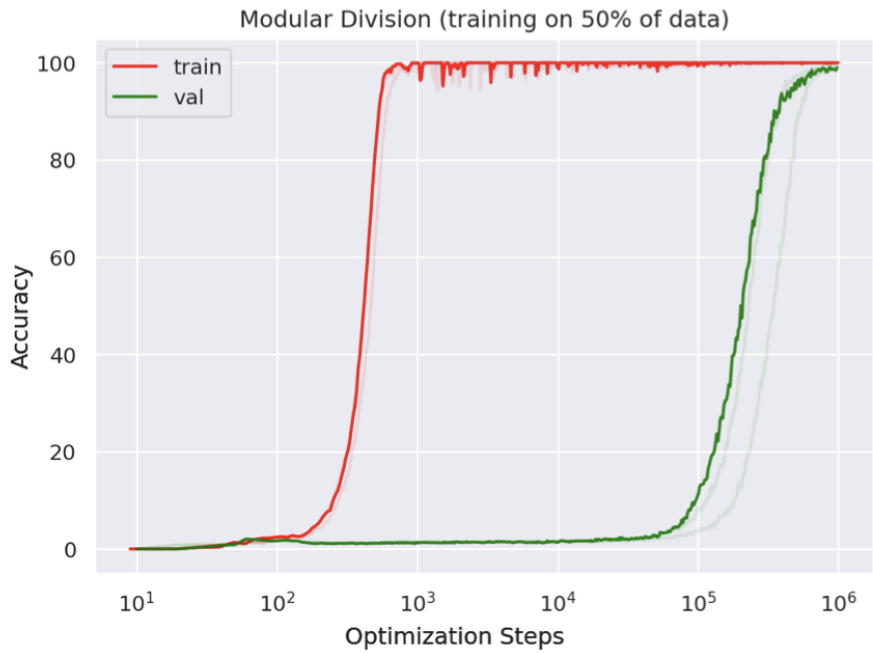
[Power, Burda, Edwards+ '22], [Liu, Kitouni, Nolte+ '22], ...

Grokking phenomenon

Grokking: Delayed generalization

- Training error drops quickly
- Test error stays high for long time
- Suddenly: test error drops

Example: Modular division task, trained on transformer architecture.



- Red: train accuracy
- Green: validation accuracy
- Gap of $\sim 10^3\times$ in steps

[Power, Burda, Edwards+ '22], [Liu, Kitouni, Nolte+ '22], ...

Demystifying grokking

Our results:

Grokking occurs
when and only when δ is **moderately above** δ_{NN}

- **Moderately above threshold:**
  - Overfit the training data in the first stage of training
  - Informative Hessian outliers in the second stage of training $\Rightarrow$ Grokking
- **Far above threshold** ($\delta \gg \delta_{\mathsf{NN}}$)**:**
  - landscape concentrates, no generalization happens

Demystifying grokking

Our results:

Grokking occurs
when and only when δ is **moderately above** δ_{NN}

- **Moderately above threshold**:
  - Overfit the training data in the first stage of training
  - Informative Hessian outlier in the second stage of training $\Rightarrow$ **Grokking**
- **Far above threshold** ($\delta \gg \delta_{\mathsf{NN}}$):
  - Landscape concentrates, no generalization gap

Demystifying grokking

Our results:

Grokking occurs
when and only when δ is **moderately above** δ_{NN}

- **Moderately above threshold**:
  - Overfit the training data in the first stage of training
  - Informative Hessian outlier in the second stage of training $\Rightarrow$ **Grokking**
- **Far above threshold** ($\delta \gg \delta_{\mathsf{NN}}$):
  - Landscape concentrates, no generalization gap

Demystifying grokking

Our results:

Grokking occurs
when and only when δ is **moderately above** δ_{NN}

- **Moderately above threshold**:
  - Overfit the training data in the first stage of training
  - Informative Hessian outlier in the second stage of training $\Rightarrow$ **Grokking**
- **Far above threshold** ($\delta \gg \delta_{\mathsf{NN}}$):
  - Landscape concentrates, no generalization gap

Demystifying grokking

Our results:

Grokking occurs
when and only when δ is **moderately above** δ_{NN}

- **Moderately above threshold**:
  - Overfit the training data in the first stage of training
  - Informative Hessian outlier in the second stage of training $\Rightarrow$ **Grokking**
- **Far above threshold** ($\delta \gg \delta_{\mathsf{NN}}$):
  - Landscape concentrates, no generalization gap

Demystifying grokking

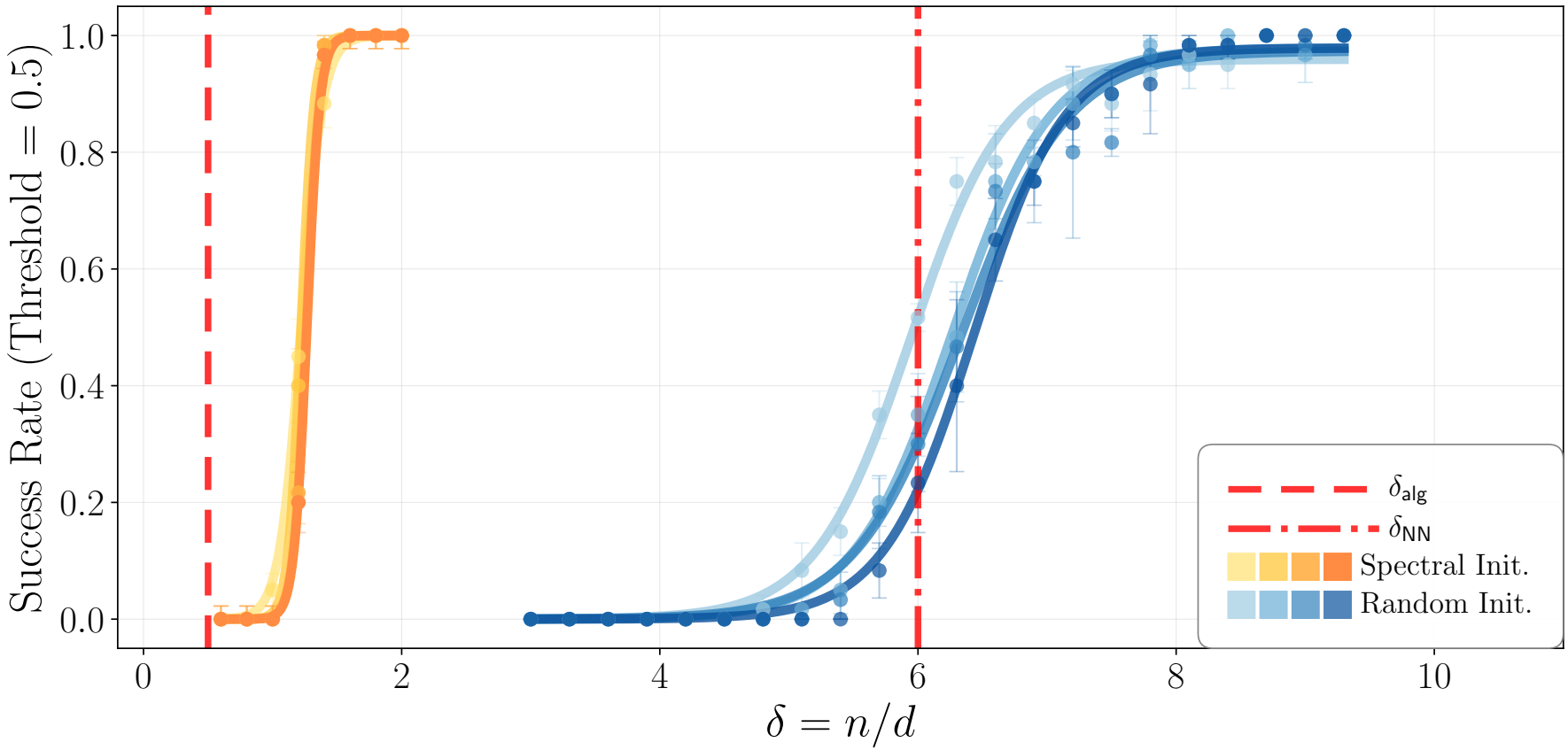
Our results:

Grokking occurs
when and only when δ is **moderately above** δ_{NN}

- **Moderately above threshold**:
  - Overfit the training data in the first stage of training
  - Informative Hessian outlier in the second stage of training $\Rightarrow$ **Grokking**
- **Far above threshold** ($\delta \gg \delta_{\mathsf{NN}}$):
  - Landscape concentrates, no generalization gap

Numerical Illustrations

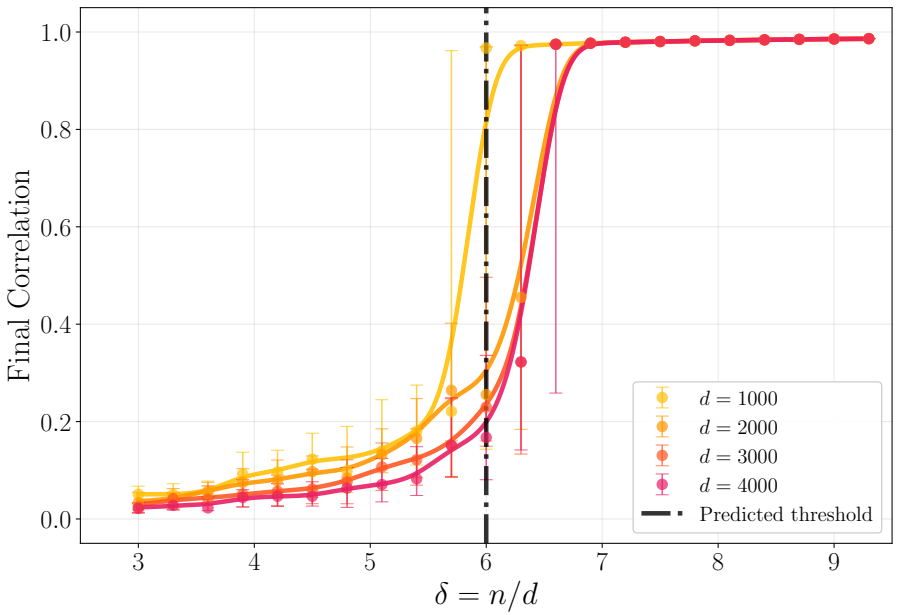
Experiments: Phase transition for feature learning



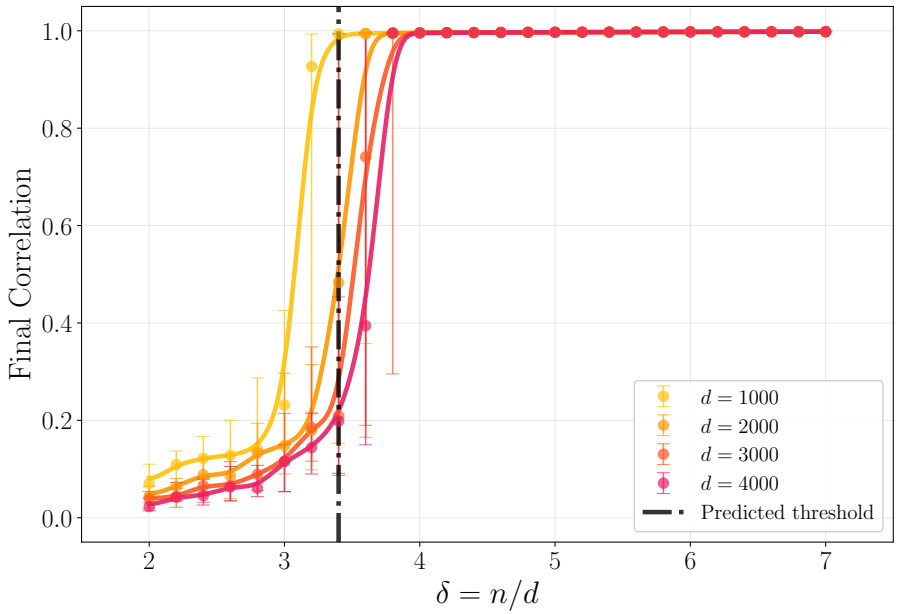
Single-neuron GeLU network learning phase retrieval ($\mathsf{h}(z) = z^2$).
Success = correlation with θ_* exceeds $1/2$.

Effect of activations

GeLU **activation**



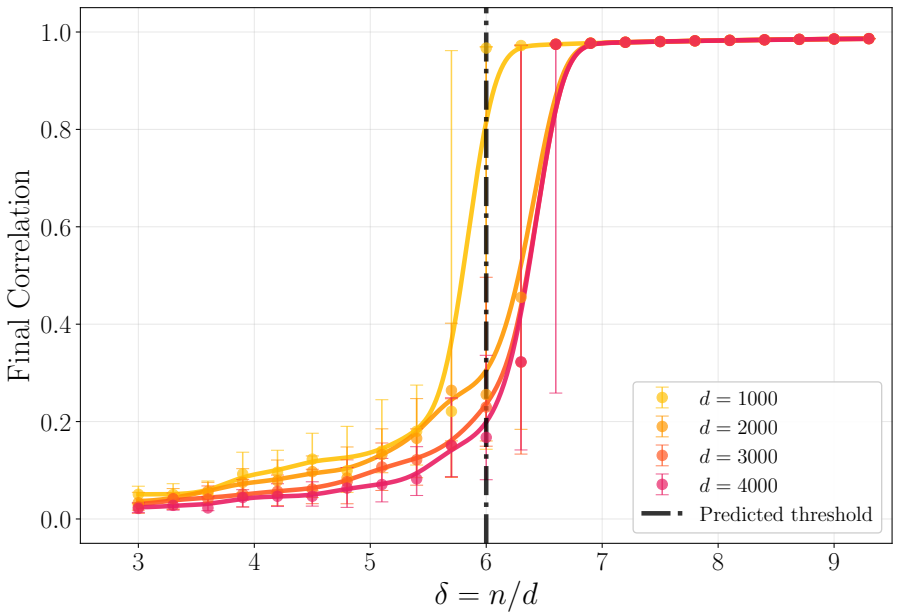
Quadratic **activation**



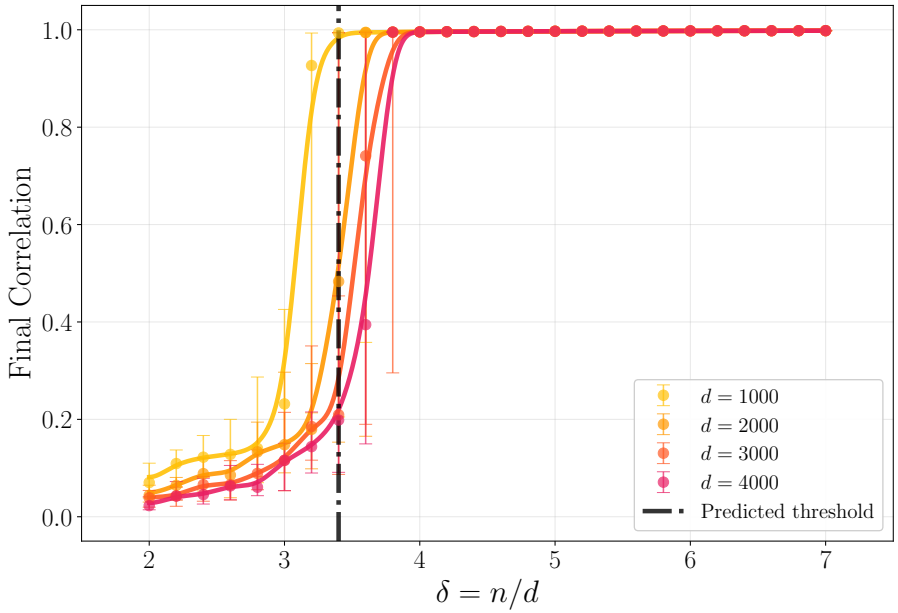
- Different activations $\Rightarrow$ different computed δ_{NN}
- Theoretical threshold matches empirical phase transition

Effect of activations

GeLU **activation**

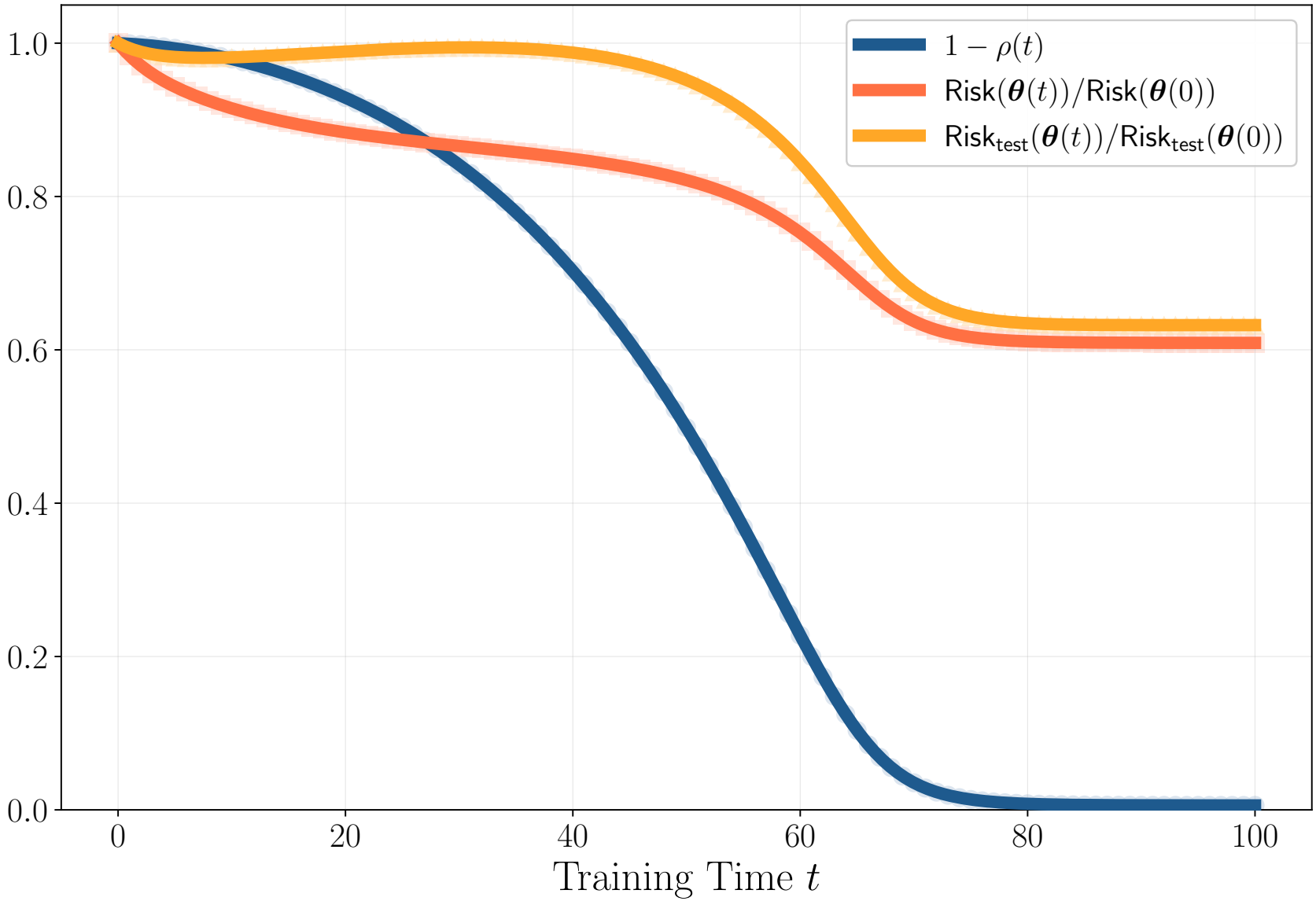


Quadratic **activation**



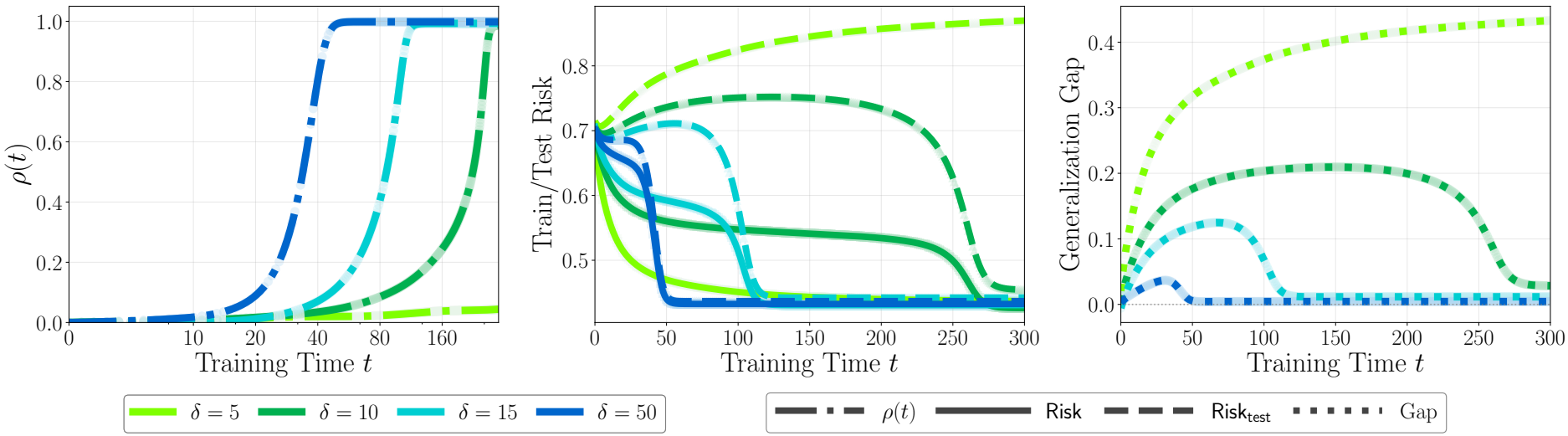
- Different activations $\Rightarrow$ different computed δ_{NN}
- Theoretical threshold matches empirical phase transition

Experiment: Grokking phenomenon



GeLU neuron, phase retrieval, $\mathsf{d} = 5000$, $\delta = 17.5$ (above $\delta_{\mathsf{NN}} \approx 6$).

Experiment: Grokking phenomenon



GeLU neuron, phase retrieval, $\mathrm{d} = 5000$, $\delta = 5, 10, 15, 50$.

Formal results and Outline of the Proofs

Structural decomposition: Insight

The latent space $\text{span}(\boldsymbol{\Theta}_*)$ decomposes into two parts:

Easy subspace

- Learned from **first-order** info of $\mathrm{h}(\cdot)$
- Gradient descent can pick up linear signal

Hard subspace

- **No first-order** info from $\mathrm{h}(\cdot)$
- Requires at least 2nd-order info to learn!

Structural decomposition: Insight

The latent space $\text{span}(\boldsymbol{\Theta}_*)$ decomposes into two parts:

Easy subspace

- Learned from **first-order** info of $\mathbf{h}(\cdot)$
- Gradient descent can pick up linear signal

Hard subspace

- **No first-order** info from $\mathbf{h}(\cdot)$
- Requires at least 2nd-order info to learn!

Structural decomposition: Insight

The latent space span($\boldsymbol{\Theta}_*$) decomposes into two parts:

Easy subspace

- Learned from **first-order** info of $\mathbf{h}(\cdot)$
- Gradient descent can pick up linear signal

Hard subspace

- **No first-order** info from $\mathbf{h}(\cdot)$
- Requires at least 2nd-order info to learn!

Structural decomposition: Formal definition

Hard subspace: $U \in \mathcal{O}(k, k')$ is a hard orthonormal set if

$$\mathbb{E}\big[\mathcal{T}(y, P_U^{\perp} z, \xi) P_U z\big] = 0, \quad \forall \text{ measurable } \mathcal{T} : \mathbb{R}^{k+2} \to \mathbb{R}$$

where $\xi \sim N(0, 1)$ is independent of (y, z) (recall $y = h(z, \varepsilon)$).

The hard subspace $\mathcal{U}_H := \operatorname{span}(U_H^*)$ is the span of the **maximal hard orthonormal set**.

$\mathcal{U}_E := \mathcal{U}_H^{\perp}$

Equivalent characterization:

$$U \text{ is hard} \quad \Leftrightarrow \quad \mathbb{E}[P_U z \mid y, P_U^{\perp} z] = 0$$

Structural decomposition: Formal definition

Hard subspace: $U \in \mathcal{O}(k, k')$ is a hard orthonormal set if

$$\mathbb{E}\big[\mathcal{T}(y, P_U^{\perp} z, \xi) P_U z\big] = 0, \quad \forall \text{ measurable } \mathcal{T} : \mathbb{R}^{k+2} \to \mathbb{R}$$

where $\xi \sim N(0, 1)$ is independent of (y, z) (recall $y = h(z, \varepsilon)$).

The hard subspace $\mathcal{U}_H := \operatorname{span}(U_H^*)$ is the span of the **maximal hard orthonormal set**.

$\mathcal{U}_E := \mathcal{U}_H^{\perp}$

Equivalent characterization:

$$U \text{ is hard} \quad \Leftrightarrow \quad \mathbb{E}[P_U z \mid y, P_U^{\perp} z] = 0$$

Structural decomposition: Formal definition

Hard subspace: $U \in \mathcal{O}(k, k')$ is a hard orthonormal set if

$$\mathbb{E}\big[\mathcal{T}(y, P_U^{\perp} z, \xi) P_U z\big] = 0, \quad \forall \text{ measurable } \mathcal{T} : \mathbb{R}^{k+2} \to \mathbb{R}$$

where $\xi \sim N(0, 1)$ is independent of (y, z) (recall $y = h(z, \varepsilon)$).

The hard subspace $\mathfrak{U}_H := \operatorname{span}(U_H^*)$ is the span of the **maximal hard orthonormal set**.

$\mathfrak{U}_E := \mathfrak{U}_H^{\perp}$

Equivalent characterization:

$$U \text{ is hard} \quad \Leftrightarrow \quad \mathbb{E}[P_U z \mid y, P_U^{\perp} z] = 0$$

Structural decomposition: Formal definition

Hard subspace: $U \in \mathcal{O}(k, k')$ is a hard orthonormal set if

$$\mathbb{E}\big[\mathcal{T}(y, P_U^{\perp} z, \xi) P_U z\big] = 0, \quad \forall \text{ measurable } \mathcal{T} : \mathbb{R}^{k+2} \to \mathbb{R}$$

where $\xi \sim N(0, 1)$ is independent of (y, z) (recall $y = h(z, \varepsilon)$).

The hard subspace $\mathfrak{U}_H := \operatorname{span}(U_H^*)$ is the span of the **maximal hard orthonormal set**.

$\mathfrak{U}_E := \mathfrak{U}_H^{\perp}$

Equivalent characterization:

$$U \text{ is hard} \quad \Leftrightarrow \quad \mathbb{E}[P_U z \mid y, P_U^{\perp} z] = 0$$

Examples

Setup: $k = 2$, $y = h(z_1, z_2)$ with h symmetric: $h(z_1, z_2) = h(z_2, z_1)$

Decomposition:

- **Easy direction:** $\mathbf{u}_E = \frac{1}{\sqrt{2}}(+1, +1)^\top$
  - $z_1 + z_2$ has linear signal from y
- **Hard direction:** $\mathbf{u}_H = \frac{1}{\sqrt{2}}(+1, -1)^\top$
  - $z_1 - z_2$ has no linear signal (symmetry!)
  - $\mathbb{E}[z_1 - z_2 \mid y, z_1 + z_2] = 0$

Other examples:

- Single-index with even $h \Rightarrow$ the only direction is hard
- Two-layer NN with even activation $\Rightarrow$ all directions are hard

Examples

Setup: $k = 2$, $y = h(z_1, z_2)$ with h symmetric: $h(z_1, z_2) = h(z_2, z_1)$

Decomposition:

- **Easy direction:** $\mathbf{u}_E = \frac{1}{\sqrt{2}}(+1, +1)^\top$
  - $z_1 + z_2$ has linear signal from y
- **Hard direction:** $\mathbf{u}_H = \frac{1}{\sqrt{2}}(+1, -1)^\top$
  - $z_1 - z_2$ has no linear signal (symmetry!)
  - $\mathbb{E}[z_1 - z_2 \mid y, z_1 + z_2] = 0$

Other examples:

- Single-index with even $h \Rightarrow$ the only direction is hard
- Two-layer NN with even activation $\Rightarrow$ all directions are hard

Examples

Setup: $k = 2$, $y = h(z_1, z_2)$ with h symmetric: $h(z_1, z_2) = h(z_2, z_1)$

Decomposition:

- **Easy direction:** $\mathbf{u}_E = \frac{1}{\sqrt{2}}(+1, +1)^\top$
  - $z_1 + z_2$ has linear signal from y
- **Hard direction:** $\mathbf{u}_H = \frac{1}{\sqrt{2}}(+1, -1)^\top$
  - $z_1 - z_2$ has no linear signal (symmetry!)
  - $\mathbb{E}[z_1 - z_2 \mid y, z_1 + z_2] = 0$

Other examples:

- Single-index with even $h \Rightarrow$ the only direction is hard
- Two-layer NN with even activation $\Rightarrow$ all directions are hard

Examples

Setup: $k = 2$, $y = h(z_1, z_2)$ with h symmetric: $h(z_1, z_2) = h(z_2, z_1)$

Decomposition:

- **Easy direction:** $\mathbf{u}_E = \frac{1}{\sqrt{2}}(+1, +1)^\top$
  - $z_1 + z_2$ has linear signal from y
- **Hard direction:** $\mathbf{u}_H = \frac{1}{\sqrt{2}}(+1, -1)^\top$
  - $z_1 - z_2$ has no linear signal (symmetry!)
  - $\mathbb{E}[z_1 - z_2 \mid y, z_1 + z_2] = 0$

Other examples:

- Single-index with even $h \Rightarrow$ the only direction is hard
- Two-layer NN with even activation $\Rightarrow$ all directions are hard

Hard directions: Not learned in $O(1)$ time

Result: For any fixed t

$$\operatorname*{p-lim}_{n,d\to\infty} \Theta(t)^\top \Theta_* P_{U_H^*} = 0$$

More general: Any estimator computed from $O(1)$ gradient steps is uncorrelated with hard directions.

Tool: Dynamical Mean Field Theory (DMFT)[5]

- Exact characterization of GD in the limit $n, d \to \infty$
- Constructs low-dimensional random vectors $(V(t), \Theta(t))$ that characterizes the dynamics

[5] Explain later. [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

Hard directions: Not learned in $O(1)$ time

Result: For any fixed t

$$\operatorname*{p-lim}_{n,d\to\infty} \Theta(t)^\top \Theta_* P_{U_H^*} = 0$$

More general: Any estimator computed from $O(1)$ gradient steps is uncorrelated with hard directions.

Tool: Dynamical Mean Field Theory (DMFT)[5]

- Exact characterization of GD in the limit $n, d \to \infty$
- Constructs low-dimensional random vectors $(V(t), \Theta(t))$ that characterizes the dynamics

[5] Explain later. [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

Hard directions: Not learned in $O(1)$ time

Result: For any fixed t

$$\operatorname*{p-lim}_{n,d\to\infty} \boldsymbol{\Theta}(t)^\top \boldsymbol{\Theta}_* P_{U_H^*} = 0$$

More general: Any estimator computed from $O(1)$ gradient steps is uncorrelated with hard directions.

Tool: Dynamical Mean Field Theory (DMFT)[5]

- Exact characterization of GD in the limit $n, d \to \infty$
- Constructs low-dimensional random vectors $(V(t), \Theta(t))$ that characterizes the dynamics

[5] Explain later. [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

Easy directions: Learned in $O(1)$ time

The network weakly learns the easy subspace within $O(1)$ iterations[6]:

There exists $\varepsilon_0 > 0$, $t = O(1)$ such that

$$\lim_{n,d\to\infty} \mathbb{P}\left(\left|\left|\boldsymbol{\Theta}(t)^\top \boldsymbol{\Theta}_* U_E^*\right|\right| \geq \varepsilon_0\right) = 1$$

Intuition:

- Easy directions have linear signal in the gradient
- Standard gradient descent picks up this signal

The learning bottleneck is the hard directions.

[6] [Dandi, Troiani, Arnaboldi, Pesce, Zdeborová, Krzakala '24]

Easy directions: Learned in $O(1)$ time

The network weakly learns the easy subspace within $O(1)$ iterations[6]:

There exists $\varepsilon_0 > 0$, $t = O(1)$ such that

$$\lim_{n,d\to\infty} \mathbb{P}\left(\left|\left|\boldsymbol{\Theta}(t)^\top \boldsymbol{\Theta}_* U_E^*\right|\right| \geq \varepsilon_0\right) = 1$$

Intuition:

- Easy directions have linear signal in the gradient
- Standard gradient descent picks up this signal

The learning bottleneck is the hard directions.

[6] [Dandi, Troiani, Arnaboldi, Pesce, Zdeborová, Krzakala '24]

Easy directions: Learned in $O(1)$ time

The network weakly learns the easy subspace within $O(1)$ iterations[6]:

There exists $\varepsilon_0 > 0$, $t = O(1)$ such that

$$\lim_{n,d\to\infty} \mathbb{P}\left(\left|\left|\boldsymbol{\Theta}(t)^\top \boldsymbol{\Theta}_* U_E^*\right|\right| \geq \varepsilon_0\right) = 1$$

Intuition:

- Easy directions have linear signal in the gradient
- Standard gradient descent picks up this signal

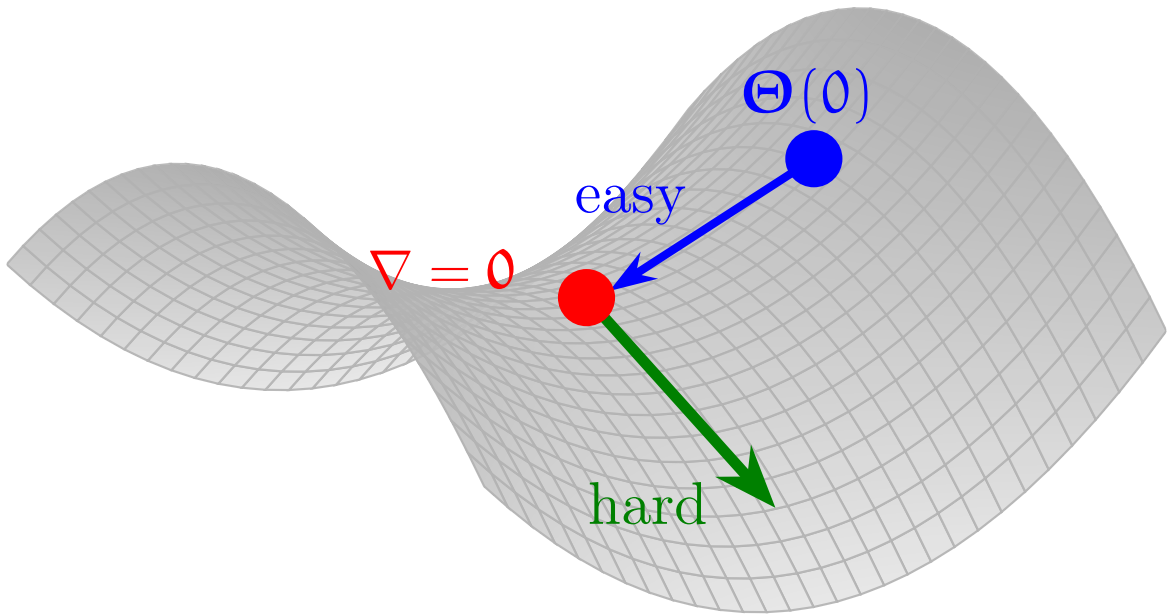
The learning bottleneck is the hard directions.

[6] [Dandi, Troiani, Arnaboldi, Pesce, Zdeborová, Krzakala '24]

Learning hard directions: The problem

Problem: No linear signal for hard directions

$\Rightarrow$ First-order methods approach a **saddle point**.



Learning hard directions: The solution

Solution: **Use second-order information to escape saddle**

- Study the (rescaled) Hessian

$$\mathbf{H}(t) := m\nabla^2 \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

  along GD trajectory
- Look for **negative eigenvalues** of the Hessian with eigenvectors aligned with hard subspace
- Such eigenvectors directions enable **escape from saddle** $\Rightarrow$ learning those hard directions

Question: When does the Hessian have **informative descent directions** after $O(1)$ iterations?

Learning hard directions: The solution

Solution: **Use second-order information to escape saddle**

- Study the (rescaled) Hessian

$$\mathbf{H}(t) := m\nabla^2 \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

along GD trajectory

- Look for **negative eigenvalues** of the Hessian with eigenvectors aligned with hard subspace
- Such eigenvectors directions enable **escape from saddle** $\Rightarrow$ learning those hard directions

Question: When does the Hessian have **informative descent directions** after $O(1)$ iterations?

Learning hard directions: The solution

Solution: **Use second-order information to escape saddle**

- Study the (rescaled) Hessian

$$\mathbf{H}(t) := m\nabla^2 \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

  along GD trajectory
- Look for **negative eigenvalues** of the Hessian with eigenvectors aligned with hard subspace
- Such eigenvectors directions enable **escape from saddle** $\Rightarrow$ learning those hard directions

Question: When does the Hessian have **informative descent directions** after $O(1)$ iterations?

Learning hard directions: The solution

Solution: **Use second-order information to escape saddle**

- Study the (rescaled) Hessian

$$\mathbf{H}(t) := m\nabla^2 \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

  along GD trajectory
- Look for **negative eigenvalues** of the Hessian with eigenvectors aligned with hard subspace
- Such eigenvectors directions enable **escape from saddle** $\Rightarrow$ learning those hard directions

Question: When does the Hessian have **informative descent directions** after $O(1)$ iterations?

Learning hard directions: The solution

Solution: **Use second-order information to escape saddle**

- Study the (rescaled) Hessian

$$\mathbf{H}(t) := m\nabla^2 \mathsf{Risk}(\boldsymbol{\Theta}(t))$$

  along GD trajectory
- Look for **negative eigenvalues** of the Hessian with eigenvectors aligned with hard subspace
- Such eigenvectors directions enable **escape from saddle** $\Rightarrow$ learning those hard directions

Question: When does the Hessian have **informative descent directions** after $O(1)$ iterations?

Hessian structure

Case $m = 1$ (single neuron): $\mathbf{H}(t) \in \mathbb{R}^{d \times d}$

$$\mathbf{H}(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \theta(t)^\top x_i) x_i x_i^\top$$

where
$g(y, z) = a^2 \ell''(y, a\sigma(z + b))\sigma'(z + b)^2 + a\ell'(y, a\sigma(z + b))\sigma''(z + b)$

Case $m \gg 1$ (wide network): $\mathbf{H}(t) \in \mathbb{R}^{md \times md}$

- Block structure with m^2 blocks of size $d \times d$
- For convex loss: well-approximated by block-diagonal $\mathbf{H}_{\text{diag}}(t)$
- Each block $a_j \mathbf{H}_j(t)$ where $\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \Theta(t)^\top x_i; j) \cdot x_i x_i^\top$
  $g(y, \Theta^\top x; j) = \ell'(y, f_\Theta(x)) \cdot \sigma''(\theta_j^\top x + b_j)$

$\Rightarrow$ Both cases reduce to analyzing **spiked random matrix** with training-dependent weights.

Hessian structure

Case $m = 1$ (single neuron): $\mathbf{H}(t) \in \mathbb{R}^{d \times d}$

$$\mathbf{H}(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \theta(t)^\top x_i) x_i x_i^\top$$

where
$g(y, z) = a^2 \ell''(y, a\sigma(z + b))\sigma'(z + b)^2 + a\ell'(y, a\sigma(z + b))\sigma''(z + b)$

Case $m \gg 1$ (wide network): $\mathbf{H}(t) \in \mathbb{R}^{md \times md}$

- Block structure with m^2 blocks of size $d \times d$
- For convex loss: well-approximated by block-diagonal $\mathbf{H}_{\text{diag}}(t)$
- Each block $a_j \mathbf{H}_j(t)$ where $\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \Theta(t)^\top x_i; j) \cdot x_i x_i^\top$
  $g(y, \Theta^\top x; j) = \ell'(y, f_\Theta(x)) \cdot \sigma''(\theta_j^\top x + b_j)$

$\Rightarrow$ Both cases reduce to analyzing **spiked random matrix** with training-dependent weights.

Hessian structure

Case $m = 1$ (single neuron): $\mathbf{H}(t) \in \mathbb{R}^{d \times d}$

$$\mathbf{H}(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \theta(t)^\top x_i) x_i x_i^\top$$

where
$g(y, z) = a^2 \ell''(y, a\sigma(z + b))\sigma'(z + b)^2 + a\ell'(y, a\sigma(z + b))\sigma''(z + b)$

Case $m \gg 1$ (wide network): $\mathbf{H}(t) \in \mathbb{R}^{md \times md}$

- Block structure with m^2 blocks of size $d \times d$
- For convex loss: well-approximated by block-diagonal $\mathbf{H}_{\text{diag}}(t)$
- Each block $a_j \mathbf{H}_j(t)$ where $\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \Theta(t)^\top x_i; j) \cdot x_i x_i^\top$
  $g(y, \Theta^\top x; j) = \ell'(y, f_\Theta(x)) \cdot \sigma''(\theta_j^\top x + b_j)$

$\Rightarrow$ Both cases reduce to analyzing **spiked random matrix** with training-dependent weights.

Main result: Hessian phase transition

(Stated for $m \gg 1$; similar result holds for $m = 1$)

Results (informal): Under the propotional asymptotic, there exist thresholds δ_j^* for each block j:

If $\delta > \delta_j^*$: For t large enough,

- $H_j(t)$ has a **negative outlier eigenvalue** below the bulk
- The corresponding eigenvector is **correlated with hard subspace**

If $\delta < \delta_j^*$: For all large enough t,

- No eigenvector of $H_j(t)$ is correlated with hard subspace

$\Rightarrow \delta_{NN} = \min_j \delta_j^*$ is the threshold for feature learning!

Main result: Hessian phase transition

(Stated for $m \gg 1$; similar result holds for $m = 1$)

Results (informal): Under the propotional asymptotic, there exist thresholds δ_j^* for each block j:

If $\delta > \delta_j^*$: For t large enough,

- $\mathbf{H}_j(t)$ has a **negative outlier eigenvalue** below the bulk
- The corresponding eigenvector is **correlated with hard subspace**

If $\delta < \delta_j^*$: For all large enough t,

- No eigenvector of $\mathbf{H}_j(t)$ is correlated with hard subspace

$\Rightarrow \delta_{NN} = \min_j \delta_j^*$ is the threshold for feature learning!

Main result: Hessian phase transition

(Stated for $m \gg 1$; similar result holds for $m = 1$)

Results (informal): Under the propotional asymptotic, there exist thresholds δ_j^* for each block j:

If $\delta > \delta_j^*$: For t large enough,

- $\mathbf{H}_j(t)$ has a **negative outlier eigenvalue** below the bulk
- The corresponding eigenvector is **correlated with hard subspace**

If $\delta < \delta_j^*$: For all large enough t,

- No eigenvector of $\mathbf{H}_j(t)$ is correlated with hard subspace

$\Rightarrow \delta_{NN} = \min_j \delta_j^*$ is the threshold for feature learning!

Main result: Hessian phase transition

(Stated for $m \gg 1$; similar result holds for $m = 1$)

Results (informal): Under the propotional asymptotic, there exist thresholds δ_j^* for each block j:

If $\delta > \delta_j^*$: For t large enough,

- $\mathbf{H}_j(t)$ has a **negative outlier eigenvalue** below the bulk
- The corresponding eigenvector is **correlated with hard subspace**

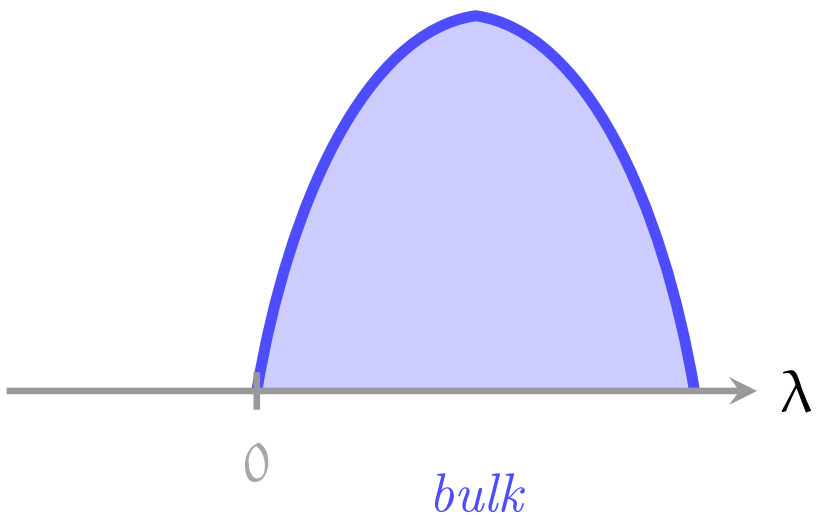
If $\delta < \delta_j^*$: For all large enough t,

- No eigenvector of $\mathbf{H}_j(t)$ is correlated with hard subspace

$\Rightarrow \delta_{NN} = \min_j \delta_j^*$ is the threshold for feature learning!

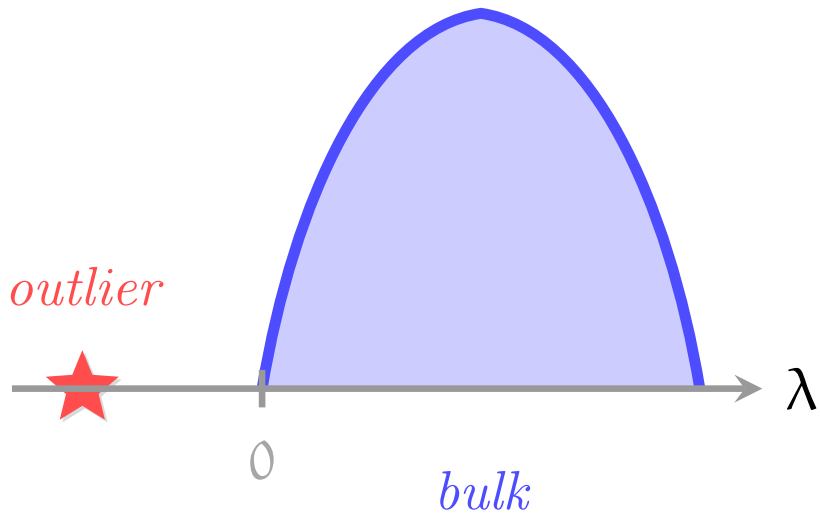
Spectrum of the Hessian

$\delta < \delta_{NN}$



No informative direction

$\delta > \delta_{NN}$



Descent direction → hard subspace!

DMFT: Evolution equations

Next, we state the formula for computing δ_{NN}.
In order to do so, we need to introduce **Dynamical Mean Field Theory (DMFT)** [7].

Goal: Characterize gradient descent dynamics as $n, d \to \infty$ via low-dimensional random variables.

$$(V(t), V_*) \in \mathbb{R}^m \times \mathbb{R}^k$$

$$V(t) = W(t) - \frac{1}{\delta} \sum_{s=0}^{t-1} R_\theta(t,s) F(V(s), V_*, \varepsilon), \quad W \sim \mathsf{GP}(0, C_\theta)$$

$$\Theta(t) \in \mathbb{R}^m$$

$$\Theta(t+1) = \Theta(t) - \eta \sum_{s=0}^{t} R_\ell(t,s)\Theta(s) - \eta R_\ell(t,*)\Theta_* + \eta Q(t), \quad Q \sim \mathsf{GP}(0, C_\ell/\delta)$$

Gradient function: $F_j(V, V_*, \varepsilon) = (\eta/m) a_j \ell'(h(V_*, \varepsilon), f(V)) \sigma'(V_j + b_j)$

[7] [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

DMFT: Evolution equations

Next, we state the formula for computing δ_{NN}.
In order to do so, we need to introduce **Dynamical Mean Field Theory (DMFT)** [7].
Goal: Characterize gradient descent dynamics as $n, d \to \infty$ via low-dimensional random variables.

$$(V(t), V_*) \in \mathbb{R}^m \times \mathbb{R}^k$$

$$V(t) = W(t) - \frac{1}{\delta} \sum_{s=0}^{t-1} R_\theta(t,s) F(V(s), V_*, \varepsilon), \quad W \sim \mathsf{GP}(0, C_\theta)$$

$$\Theta(t) \in \mathbb{R}^m$$

$$\Theta(t+1) = \Theta(t) - \eta \sum_{s=0}^{t} R_\ell(t,s)\Theta(s) - \eta R_\ell(t,*)\Theta_* + \eta Q(t), \quad Q \sim \mathsf{GP}(0, C_\ell/\delta)$$

Gradient function: $F_j(V, V_*, \varepsilon) = (\eta/m) a_j \ell'(h(V_*, \varepsilon), f(V)) \sigma'(V_j + b_j)$

[7] [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

DMFT: Evolution equations

Next, we state the formula for computing δ_{NN}.
In order to do so, we need to introduce **Dynamical Mean Field Theory (DMFT)** [7].
Goal: Characterize gradient descent dynamics as $n, d \to \infty$ via low-dimensional random variables.

$(V(t), V_*) \in \mathbb{R}^m \times \mathbb{R}^k$

$$V(t) = W(t) - \frac{1}{\delta} \sum_{s=0}^{t-1} R_\theta(t,s) F(V(s), V_*, \varepsilon), \quad W \sim \mathsf{GP}(0, C_\theta)$$

$\Theta(t) \in \mathbb{R}^m$

$$\Theta(t+1) = \Theta(t) - \eta \sum_{s=0}^{t} R_\ell(t,s)\Theta(s) - \eta R_\ell(t,*)\Theta_* + \eta Q(t), \quad Q \sim \mathsf{GP}(0, C_\ell/\delta)$$

Gradient function: $F_j(V, V_*, \varepsilon) = (\eta/m) a_j \ell'(h(V_*, \varepsilon), f(V)) \sigma'(V_j + b_j)$

[7] [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

DMFT: Evolution equations

Next, we state the formula for computing δ_{NN}.
In order to do so, we need to introduce **Dynamical Mean Field Theory (DMFT)** [7].
Goal: Characterize gradient descent dynamics as $n, d \to \infty$ via low-dimensional random variables.

$(V(t), V_*) \in \mathbb{R}^m \times \mathbb{R}^k$

$$V(t) = W(t) - \frac{1}{\delta} \sum_{s=0}^{t-1} R_\theta(t,s) F(V(s), V_*, \varepsilon), \quad W \sim \mathsf{GP}(0, C_\theta)$$

$\Theta(t) \in \mathbb{R}^m$

$$\Theta(t+1) = \Theta(t) - \eta \sum_{s=0}^{t} R_\ell(t,s)\Theta(s) - \eta R_\ell(t,*)\Theta_* + \eta Q(t), \quad Q \sim \mathsf{GP}(0, C_\ell/\delta)$$

Gradient function: $F_j(V, V_*, \varepsilon) = (\eta/m) a_j \ell'(h(V_*, \varepsilon), f(V)) \sigma'(V_j + b_j)$

[7] [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

DMFT: Evolution equations

Next, we state the formula for computing δ_{NN}.
In order to do so, we need to introduce **Dynamical Mean Field Theory (DMFT)** [7].
Goal: Characterize gradient descent dynamics as $n, d \to \infty$ via low-dimensional random variables.

$(V(t), V_*) \in \mathbb{R}^m \times \mathbb{R}^k$

$$V(t) = W(t) - \frac{1}{\delta} \sum_{s=0}^{t-1} R_\theta(t,s) F(V(s), V_*, \varepsilon), \quad W \sim \mathsf{GP}(0, C_\theta)$$

$\Theta(t) \in \mathbb{R}^m$

$$\Theta(t+1) = \Theta(t) - \eta \sum_{s=0}^{t} R_\ell(t,s)\Theta(s) - \eta R_\ell(t,*)\Theta_* + \eta Q(t), \quad Q \sim \mathsf{GP}(0, C_\ell/\delta)$$

Gradient function: $F_j(V, V_*, \varepsilon) = (\eta/m) a_j \ell'(h(V_*, \varepsilon), f(V)) \sigma'(V_j + b_j)$

[7] [Sompolinsky, Zippelius '81, '82; Celentano, Montanari, Wu '20; Celentano, Cheng, Montanari '21]

DMFT: Evolution equations

Covariance kernels:

$$C_\theta(t,s) = \mathbb{E}[\Theta(t)\Theta(s)^\top], \quad C_\theta(t,*) = \mathbb{E}[\Theta(t)\Theta_*^\top]$$
$$C_\ell(t,s) = \frac{1}{n^2}\mathbb{E}[F(V(t),V_*,\varepsilon)F(V(s),V_*,\varepsilon)^\top]$$

Response kernels:

$$R_\theta(t,s) = \frac{1}{n}\mathbb{E}\left[\frac{\partial\Theta(t)}{\partial Q(s)}\right]$$
$$R_\ell(t,s) = \frac{1}{n}\mathbb{E}\left[\frac{\partial F(V(t),V_*,\varepsilon)}{\partial W(s)}\right], \quad R_\ell(t,*) = \frac{1}{n}\mathbb{E}\left[\frac{\partial F(V(t),V_*,\varepsilon)}{\partial V_*}\right]$$

Initialization:

$$\Theta(0) \sim N(0, I_m) \text{ independent of } \Theta_* \sim N(0, I_k)$$

DMFT: Evolution equations

Covariance kernels:

$$C_\theta(t,s) = \mathbb{E}[\Theta(t)\Theta(s)^\top], \quad C_\theta(t,*) = \mathbb{E}[\Theta(t)\Theta_*^\top]$$
$$C_\ell(t,s) = \frac{1}{\eta^2}\mathbb{E}[F(V(t),V_*,\varepsilon)F(V(s),V_*,\varepsilon)^\top]$$

Response kernels:

$$R_\theta(t,s) = \frac{1}{\eta}\mathbb{E}\left[\frac{\partial\Theta(t)}{\partial Q(s)}\right]$$
$$R_\ell(t,s) = \frac{1}{\eta}\mathbb{E}\left[\frac{\partial F(V(t),V_*,\varepsilon)}{\partial W(s)}\right], \quad R_\ell(t,*) = \frac{1}{\eta}\mathbb{E}\left[\frac{\partial F(V(t),V_*,\varepsilon)}{\partial V_*}\right]$$

Initialization:

$$\Theta(0) \sim N(0, I_m) \text{ independent of } \Theta_* \sim N(0, I_k)$$

DMFT: Evolution equations

Covariance kernels:

$$C_\theta(t,s) = \mathbb{E}[\Theta(t)\Theta(s)^\top], \quad C_\theta(t,*) = \mathbb{E}[\Theta(t)\Theta_*^\top]$$
$$C_\ell(t,s) = \frac{1}{\eta^2}\mathbb{E}[F(V(t),V_*,\varepsilon)F(V(s),V_*,\varepsilon)^\top]$$

Response kernels:

$$R_\theta(t,s) = \frac{1}{\eta}\mathbb{E}\left[\frac{\partial\Theta(t)}{\partial Q(s)}\right]$$
$$R_\ell(t,s) = \frac{1}{\eta}\mathbb{E}\left[\frac{\partial F(V(t),V_*,\varepsilon)}{\partial W(s)}\right], \quad R_\ell(t,*) = \frac{1}{\eta}\mathbb{E}\left[\frac{\partial F(V(t),V_*,\varepsilon)}{\partial V_*}\right]$$

Initialization:

$$\Theta(0) \sim N(0, I_m) \text{ independent of } \Theta_* \sim N(0, I_k)$$

DMFT: Convergence

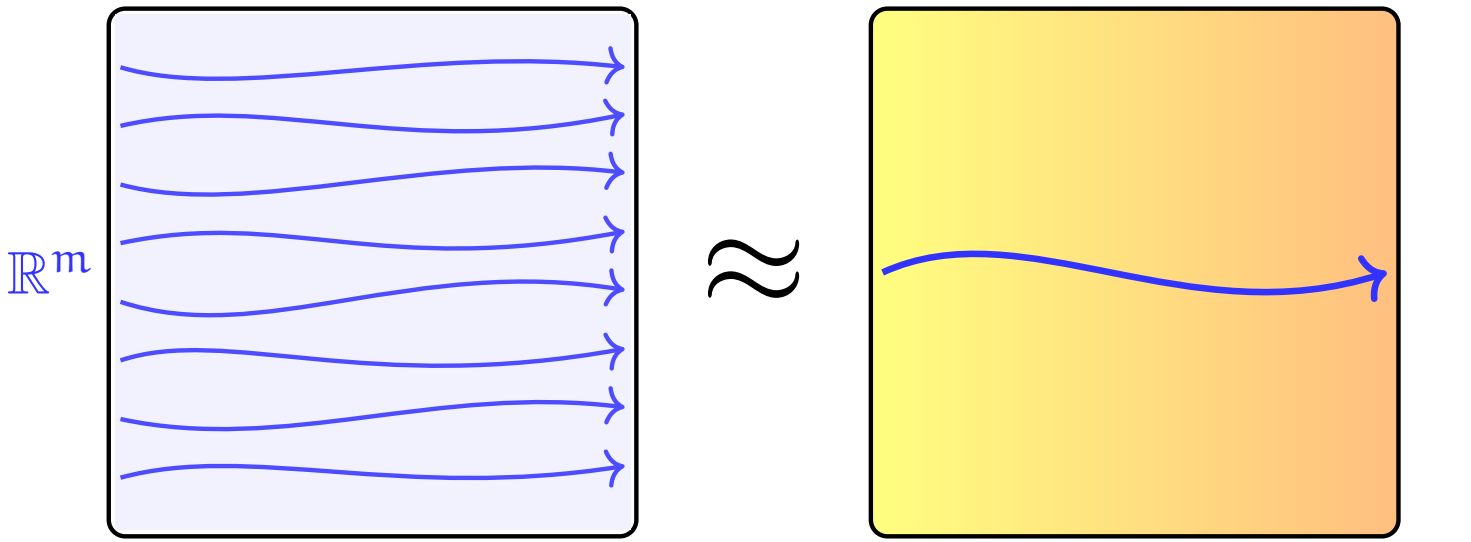
For test function ϕ,

$$\frac{1}{n}\sum_{i=1}^{n}\phi(\boldsymbol{\Theta}(t)^\top \mathbf{x}_i, \boldsymbol{\Theta}_*^\top \mathbf{x}_i, \varepsilon_i) \xrightarrow{p} \mathbb{E}\big[\phi(V(t), V_*, \varepsilon)\big]$$

$$\frac{1}{d}\sum_{i=1}^{d}\phi(\sqrt{d}\boldsymbol{\Theta}(t)_i, \sqrt{d}\boldsymbol{\Theta}_{*i}) \xrightarrow{p} \mathbb{E}\big[\phi(\Theta(t), \Theta_*)\big]$$

Remark: Formulas scale with k, m and t, **not** with n and d

DMFT: Main idea



d particles in $\mathbb{R}^m$ one part. + noise + memory

'Gaussian' approximation + Symmetries

$\Rightarrow$ Integral equations for memory + response kernels

Bulk spectrum of the Hessian

Hessian block: $\mathbf{H}_j(t) = \frac{1}{n}\sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top$

Limiting spectral distribution: $\mu_{n,d}^j(t) \Rightarrow \mu_\infty^j(t)$ with $\alpha_t^j(z)$ the Stieltjes transform of $\mu_\infty^j(t)$.

Stieltjes transform: For $z \in \mathbb{C}^+$, $\alpha_t^j(z)$ is the unique solution of

$$z + \frac{1}{\alpha_t^j(z)} = \delta \cdot \mathbb{E}\left[\frac{G_t^j}{\delta + G_t^j \alpha_t^j(z)}\right]$$

where $G_t^j := g(V(t), h(V_*, \varepsilon); j)$.

Remark: Generalized Marchenko-Pastur law despite complex dependence.

Bulk spectrum of the Hessian

Hessian block: $\mathbf{H}_j(t) = \frac{1}{n}\sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top$

Limiting spectral distribution: $\mu_{n,d}^j(t) \Rightarrow \mu_\infty^j(t)$ with $\alpha_t^j(z)$ the Stieltjes transform of $\mu_\infty^j(t)$.

Stieltjes transform: For $z \in \mathbb{C}^+$, $\alpha_t^j(z)$ is the unique solution of

$$z + \frac{1}{\alpha_t^j(z)} = \delta \cdot \mathbb{E}\left[\frac{G_t^j}{\delta + G_t^j \alpha_t^j(z)}\right]$$

where $G_t^j := g(V(t), h(V_*, \varepsilon); j)$.

Remark: Generalized Marchenko-Pastur law despite complex dependence.

Bulk spectrum of the Hessian

Hessian block: $\mathbf{H}_j(t) = \frac{1}{n}\sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top$

Limiting spectral distribution: $\mu_{n,d}^j(t) \Rightarrow \mu_\infty^j(t)$ with $\alpha_t^j(z)$ the Stieltjes transform of $\mu_\infty^j(t)$.

Stieltjes transform: For $z \in \mathbb{C}^+$, $\alpha_t^j(z)$ is the unique solution of

$$z + \frac{1}{\alpha_t^j(z)} = \delta \cdot \mathbb{E}\left[\frac{G_t^j}{\delta + G_t^j \alpha_t^j(z)}\right]$$

where $G_t^j := g(V(t), h(V_*, \varepsilon); j)$.

Remark: Generalized Marchenko-Pastur law despite complex dependence.

Bulk spectrum of the Hessian

Hessian block: $\mathbf{H}_j(t) = \frac{1}{n}\sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top$

Limiting spectral distribution: $\mu_{n,d}^j(t) \Rightarrow \mu_\infty^j(t)$ with $\alpha_t^j(z)$ the Stieltjes transform of $\mu_\infty^j(t)$.

Stieltjes transform: For $z \in \mathbb{C}^+$, $\alpha_t^j(z)$ is the unique solution of

$$z + \frac{1}{\alpha_t^j(z)} = \delta \cdot \mathbb{E}\left[\frac{G_t^j}{\delta + G_t^j \alpha_t^j(z)}\right]$$

where $G_t^j := g(V(t), h(V_*, \varepsilon); j)$.

Remark: Generalized Marchenko-Pastur law despite complex dependence.

Edge and outlier equations

Left edge: $$A_t^j := \frac{\delta}{(-\inf(\mathrm{supp}(\mathrm{Law}(G_t^j)))) \vee 0}$$

$$c_j(t) := \sup_{\alpha \in (0, A_t^j)} \left\{ -\frac{1}{\alpha} + \delta \cdot \mathbb{E}\left[\frac{G_t^j}{\delta + G_t^j \alpha} \right] \right\}$$

Outlier equation: For $z < \min(0, c_j(t))$,

$$\det\left(-z I_r + \mathbb{E}\left[\frac{\delta G_t^j}{\delta + G_t^j \alpha_t^j(z)} U_H^{*\top} V_* (U_H^{*\top} V_*)^\top \right] \right) = 0$$

Under $n/d \to \delta \in (0, \infty)$:

- Outlier eigenvalues $\longrightarrow$ solutions of outlier equation
- Corresponding eigenvectors correlated with hard subspace

Edge and outlier equations

Left edge: $$A_t^j := \frac{\delta}{(-\inf(\mathrm{supp}(\mathrm{Law}(G_t^j)))) \vee 0}$$

$$c_j(t) := \sup_{\alpha \in (0, A_t^j)} \left\{ -\frac{1}{\alpha} + \delta \cdot \mathbb{E}\left[\frac{G_t^j}{\delta + G_t^j \alpha} \right] \right\}$$

Outlier equation: For $z < \min(0, c_j(t))$,

$$\det\left(-z I_r + \mathbb{E}\left[\frac{\delta G_t^j}{\delta + G_t^j \alpha_t^j(z)} U_H^{*\top} V_* (U_H^{*\top} V_*)^\top \right] \right) = 0$$

Under $n/d \to \delta \in (0, \infty)$:

- Outlier eigenvalues $\longrightarrow$ solutions of outlier equation
- Corresponding eigenvectors correlated with hard subspace

Edge and outlier equations

Left edge: $$A_t^j := \frac{\delta}{(-\inf(\mathrm{supp}(\mathrm{Law}(G_t^j)))) \vee 0}$$

$$c_j(t) := \sup_{\alpha \in (0, A_t^j)} \left\{ -\frac{1}{\alpha} + \delta \cdot \mathbb{E}\left[\frac{G_t^j}{\delta + G_t^j \alpha} \right] \right\}$$

Outlier equation: For $z < \min(0, c_j(t))$,

$$\det\left(-z I_r + \mathbb{E}\left[\frac{\delta G_t^j}{\delta + G_t^j \alpha_t^j(z)} U_H^{*\top} V_* (U_H^{*\top} V_*)^\top \right] \right) = 0$$

Under $n/d \to \delta \in (0, \infty)$:

- Outlier eigenvalues $\longrightarrow$ solutions of outlier equation
- Corresponding eigenvectors correlated with hard subspace

Threshold definition

$$\delta_j^*(t) := \inf\{\delta > 0 : \text{outlier eq. has solution } z < \min(0, c_j(t))\}$$

$$\delta_j^*(\infty) := \lim_{t\to\infty} \delta_j^*(t)$$

Overall: $$\delta_{NN} := \min_{j\in[m]} \delta_j^*(\infty)$$

- $\delta > \delta_{NN}$: Hessian has informative descent direction $\Rightarrow$ learn hard subspace
- $\delta < \delta_{NN}$: No informative direction $\Rightarrow$ cannot learn

Remark: For $m = 1$, similar formulas hold with G_t^j replaced by $G_t = \ell''\sigma'^2 + \ell'\sigma''$.

Threshold definition

$$\delta_j^*(t) := \inf\{\delta > 0 : \text{outlier eq. has solution } z < \min(0, c_j(t))\}$$

$$\delta_j^*(\infty) := \lim_{t\to\infty} \delta_j^*(t)$$

Overall: $$\delta_{NN} := \min_{j\in[m]} \delta_j^*(\infty)$$

- $\delta > \delta_{NN}$: Hessian has informative descent direction $\Rightarrow$ learn hard subspace
- $\delta < \delta_{NN}$: No informative direction $\Rightarrow$ cannot learn

Remark: For $m = 1$, similar formulas hold with G_t^j replaced by $G_t = \ell''\sigma'^2 + \ell'\sigma''$.

Threshold definition

$$\delta_j^*(t) := \inf\{\delta > 0 : \text{outlier eq. has solution } z < \min(0, c_j(t))\}$$

$$\delta_j^*(\infty) := \lim_{t\to\infty} \delta_j^*(t)$$

Overall: $\delta_{NN} := \min_{j\in[m]} \delta_j^*(\infty)$

- $\delta > \delta_{NN}$: Hessian has informative descent direction $\Rightarrow$ learn hard subspace
- $\delta < \delta_{NN}$: No informative direction $\Rightarrow$ cannot learn

Remark: For $m = 1$, similar formulas hold with G_t^j replaced by $G_t = \ell''\sigma'^2 + \ell'\sigma''$.

Threshold definition

$$\delta_j^*(t) := \inf\{\delta > 0 : \text{outlier eq. has solution } z < \min(0, c_j(t))\}$$

$$\delta_j^*(\infty) := \lim_{t\to\infty} \delta_j^*(t)$$

Overall: $\delta_{NN} := \min_{j\in[m]} \delta_j^*(\infty)$

- $\delta > \delta_{NN}$: Hessian has informative descent direction $\Rightarrow$ learn hard subspace
- $\delta < \delta_{NN}$: No informative direction $\Rightarrow$ cannot learn

Remark: For $m = 1$, similar formulas hold with G_t^j replaced by $G_t = \ell''\sigma'^2 + \ell'\sigma''$.

Threshold definition

$$\delta_j^*(t) := \inf\{\delta > 0 : \text{outlier eq. has solution } z < \min(0, c_j(t))\}$$

$$\delta_j^*(\infty) := \lim_{t\to\infty} \delta_j^*(t)$$

Overall: $\delta_{NN} := \min_{j\in[m]} \delta_j^*(\infty)$

- $\delta > \delta_{NN}$: Hessian has informative descent direction $\Rightarrow$ learn hard subspace
- $\delta < \delta_{NN}$: No informative direction $\Rightarrow$ cannot learn

Remark: For $m = 1$, similar formulas hold with G_t^j replaced by $G_t = \ell''\sigma'^2 + \ell'\sigma''$.

Grokking explanation

Setup: (We assume for now) only hard directions to learn, $\delta > \delta_{\mathsf{NN}}$

Stage 1: $t = O(1)$

- Cannot learn hard directions (no linear signal)
- But $n/d = \delta$ is finite $\Rightarrow$ **overfits a bit training data**
- Train loss $\ll$ Test loss (generalization gap)

Stage 2: Saddle escape happens

- Since $\delta > \delta_{\mathsf{NN}}$: Hessian **develops informative outlier**
- Eventually **learns hard directions**
- Test loss drops $\Rightarrow$ **Predictable grokking!**

No grokking when $\delta \gg \delta_{\mathsf{NN}}$:

- No overfitting in Stage 1
- No generalization gap $\Rightarrow$ no delayed generalization

Grokking explanation

Setup: (We assume for now) only hard directions to learn, $\delta > \delta_{NN}$

Stage 1: $t = O(1)$

- Cannot learn hard directions (no linear signal)
- But $n/d = \delta$ is finite $\Rightarrow$ **overfits a bit training data**
- Train loss $\ll$ Test loss (generalization gap)

Stage 2: Saddle escape happens

- Since $\delta > \delta_{NN}$: Hessian **develops informative outlier**
- Eventually **learns hard directions**
- Test loss drops $\Rightarrow$ **Predictable grokking!**

No grokking when $\delta \gg \delta_{NN}$:

- No overfitting in Stage 1
- No generalization gap $\Rightarrow$ no delayed generalization

Grokking explanation

Setup: (We assume for now) only hard directions to learn, $\delta > \delta_{NN}$

Stage 1: $t = O(1)$

- Cannot learn hard directions (no linear signal)
- But $n/d = \delta$ is finite $\Rightarrow$ **overfits a bit training data**
- Train loss $\ll$ Test loss (generalization gap)

Stage 2: Saddle escape happens

- Since $\delta > \delta_{NN}$: Hessian **develops informative outlier**
- Eventually **learns hard directions**
- Test loss drops $\Rightarrow$ **Predictable grokking!**

No grokking when $\delta \gg \delta_{NN}$:

- No overfitting in Stage 1
- No generalization gap $\Rightarrow$ no delayed generalization

Grokking explanation

Setup: (We assume for now) only hard directions to learn, $\delta > \delta_{NN}$

Stage 1: $t = O(1)$

- Cannot learn hard directions (no linear signal)
- But $n/d = \delta$ is finite $\Rightarrow$ **overfits a bit training data**
- Train loss $\ll$ Test loss (generalization gap)

Stage 2: Saddle escape happens

- Since $\delta > \delta_{NN}$: Hessian **develops informative outlier**
- Eventually **learns hard directions**
- Test loss drops $\Rightarrow$ **Predictable grokking!**

No grokking when $\delta \gg \delta_{NN}$:

- No overfitting in Stage 1
- No generalization gap $\Rightarrow$ no delayed generalization

Proof techniques

Our proofs utilize three main ingredients:

1. **Dynamical Mean Field Theory (DMFT)**
   - Tracks the evolution of GD in the first stage of training
2. **Gaussian conditioning**
3. **Random Matrix Theory (RMT)**
   - The later two techniques are used for characterizing the Hessian spectrum (second stage).

▷ Proofs are mathematically involved but conceptually straightforward. Detailed proof sketch in the appendix.

Proof techniques

Our proofs utilize three main ingredients:

1. **Dynamical Mean Field Theory (DMFT)**
   - Tracks the evolution of GD in the first stage of training
2. **Gaussian conditioning**
3. **Random Matrix Theory (RMT)**
   - The later two techniques are used for characterizing the Hessian spectrum (second stage).

$\triangleright$ Proofs are mathematically involved but conceptually straightforward. Detailed proof sketch in the appendix.

Discussion

Conclusion

Main contributions:

1. **Precise threshold δ_{NN}** for feature learning (the first hard direction) via gradient descent on two-layer NNs
   - ▻ Depends on target, loss, activation, width, initialization
   - ▻ Efficiently computable from problem setup
2. **Spectral mechanism:** Learning hard directions $\Leftrightarrow$ negative outliers of Hessian aligned with hard subspace
3. $\delta_{\mathsf{NN}} > \delta_{\mathsf{alg}}$ in general; gap depends on architecture and training
4. **Grokking explanation:** Delayed generalization predictably occurs when and only when δ is moderately above δ_{NN}

Conclusion

Main contributions:

1. **Precise threshold δ_{NN}** for feature learning (the first hard direction) via gradient descent on two-layer NNs
   - Depends on target, loss, activation, width, initialization
   - Efficiently computable from problem setup
2. **Spectral mechanism:** Learning hard directions $\Leftrightarrow$ negative outliers of Hessian aligned with hard subspace
3. $\delta_{\mathsf{NN}} > \delta_{\mathsf{alg}}$ in general; gap depends on architecture and training
4. **Grokking explanation:** Delayed generalization predictably occurs when and only when δ is moderately above δ_{NN}

Conclusion

Main contributions:

1. **Precise threshold δ_{NN}** for feature learning (the first hard direction) via gradient descent on two-layer NNs
   - Depends on target, loss, activation, width, initialization
   - Efficiently computable from problem setup
2. **Spectral mechanism:** Learning hard directions $\Leftrightarrow$ negative outliers of Hessian aligned with hard subspace
3. $\delta_{\mathsf{NN}} > \delta_{\mathsf{alg}}$ in general; gap depends on architecture and training
4. **Grokking explanation:** Delayed generalization predictably occurs when and only when δ is moderately above δ_{NN}

Conclusion

Main contributions:

1. **Precise threshold δ_{NN}** for feature learning (the first hard direction) via gradient descent on two-layer NNs
   - Depends on target, loss, activation, width, initialization
   - Efficiently computable from problem setup
2. **Spectral mechanism:** Learning hard directions $\Leftrightarrow$ negative outliers of Hessian aligned with hard subspace
3. $\delta_{\mathsf{NN}} > \delta_{\mathsf{alg}}$ in general; gap depends on architecture and training
4. **Grokking explanation:** Delayed generalization predictably occurs when and only when δ is moderately above δ_{NN}

Conclusion

Main contributions:

1. **Precise threshold δ_{NN}** for feature learning (the first hard direction) via gradient descent on two-layer NNs
   - Depends on target, loss, activation, width, initialization
   - Efficiently computable from problem setup
2. **Spectral mechanism:** Learning hard directions $\Leftrightarrow$ negative outliers of Hessian aligned with hard subspace
3. $\delta_{\mathsf{NN}} > \delta_{\mathsf{alg}}$ in general; gap depends on architecture and training
4. **Grokking explanation:** Delayed generalization predictably occurs when and only when δ is moderately above δ_{NN}

Conclusion

Main contributions:

1. **Precise threshold δ_{NN}** for feature learning (the first hard direction) via gradient descent on two-layer NNs
   - Depends on target, loss, activation, width, initialization
   - Efficiently computable from problem setup
2. **Spectral mechanism:** Learning hard directions $\Leftrightarrow$ negative outliers of Hessian aligned with hard subspace
3. $\delta_{\mathsf{NN}} > \delta_{\mathsf{alg}}$ in general; gap depends on architecture and training
4. **Grokking explanation:** Delayed generalization predictably occurs when and only when δ is moderately above δ_{NN}

Conclusion

Main contributions:

1. **Precise threshold δ_{NN}** for feature learning (the first hard direction) via gradient descent on two-layer NNs
   - Depends on target, loss, activation, width, initialization
   - Efficiently computable from problem setup
2. **Spectral mechanism:** Learning hard directions $\Leftrightarrow$ negative outliers of Hessian aligned with hard subspace
3. $\delta_{\mathsf{NN}} > \delta_{\mathsf{alg}}$ in general; gap depends on architecture and training
4. **Grokking explanation:** Delayed generalization predictably occurs when and only when δ is moderately above δ_{NN}

Remarks

Multi-index models as a testbed

- Multi-index models are an excellent mathematical framework for understanding feature learning. It captures the essential challenge: learning low-dimensional structure from high-dimensional data.
- Simple enough for performing precise mathematical arguments, complex enough to rule out kernel methods and observe non-trivial phenomena observed in practice (feature learning, grokking, etc).

Beyond multi-index models

- hierarchical/compositional targets

$$y = g(p_1(\boldsymbol{x}), \ldots, p_k(\boldsymbol{x})) \text{ or } y = g_k \circ \cdots \circ g_1(\boldsymbol{x})$$

- Interesting in capturing hierarchical feature learning in deep networks

Remarks

Multi-index models as a testbed

- Multi-index models are an excellent mathematical framework for understanding feature learning. It captures the essential challenge: learning low-dimensional structure from high-dimensional data.
- Simple enough for performing precise mathematical arguments, complex enough to rule out kernel methods and observe non-trivial phenomena observed in practice (feature learning, grokking, etc).

Beyond multi-index models

- hierarchical/compositional targets

$$y = g(p_1(\boldsymbol{x}), \ldots, p_k(\boldsymbol{x})) \text{ or } y = g_k \circ \cdots \circ g_1(\boldsymbol{x})$$

- Interesting in capturing hierarchical feature learning in deep networks

Thanks for listening!

Questions?

arXiv:2602.01434

Proof sketch: Setup

(WLOG consider $m \gg 1$; the case $m = 1$ is similar)

Object of interest: Hessian diagonal block

$$H_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \Theta(t)^\top x_i; j) \cdot x_i x_i^\top = X^\top G_j(t) X$$

Intuition: y_i only depends on **few directions** $\Theta_*^\top x_i$
$\Rightarrow$ $G_j(t)$ carries low-rank informative signal $\Rightarrow$ expect outliers

Why is this hard? Not a standard spiked random matrix

- $\Theta(t)$ depends on (X, y) through GD updates
- Complex nonlinear dependence $\Rightarrow$ standard RMT does not work

Key insight: Although nonlinear, each GD update is **linear in X**

Proof sketch: Setup

(WLOG consider $m \gg 1$; the case $m = 1$ is similar)

Object of interest: Hessian diagonal block

$$\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \mathbf{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X}^\top \mathbf{G}_j(t) \mathbf{X}$$

Intuition: y_i only depends on **few directions** $\mathbf{\Theta}_*^\top \mathbf{x}_i$
$\Rightarrow$ $\mathbf{G}_j(t)$ carries low-rank informative signal $\Rightarrow$ expect outliers

Why is this hard? Not a standard spiked random matrix

- $\mathbf{\Theta}(t)$ depends on $(\mathbf{X}, \mathbf{y})$ through GD updates
- Complex nonlinear dependence $\Rightarrow$ standard RMT does not work

Key insight: Although nonlinear, each GD update is **linear in X**

Proof sketch: Setup

(WLOG consider $m \gg 1$; the case $m = 1$ is similar)

Object of interest: Hessian diagonal block

$$\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X}^\top \mathbf{G}_j(t) \mathbf{X}$$

Intuition: y_i only depends on **few directions** $\boldsymbol{\Theta}_*^\top \mathbf{x}_i$
$\Rightarrow$ $\mathbf{G}_j(t)$ carries low-rank informative signal $\Rightarrow$ expect outliers

Why is this hard? Not a standard spiked random matrix

- $\boldsymbol{\Theta}(t)$ depends on $(\mathbf{X}, \mathbf{y})$ through GD updates
- Complex nonlinear dependence $\Rightarrow$ standard RMT does not work

Key insight: Although nonlinear, each GD update is **linear in X**

Proof sketch: Setup

(WLOG consider $m \gg 1$; the case $m = 1$ is similar)

Object of interest: Hessian diagonal block

$$\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X}^\top \mathbf{G}_j(t) \mathbf{X}$$

Intuition: y_i only depends on **few directions** $\boldsymbol{\Theta}_*^\top \mathbf{x}_i$
$\Rightarrow$ $\mathbf{G}_j(t)$ carries low-rank informative signal $\Rightarrow$ expect outliers

Why is this hard? Not a standard spiked random matrix

- $\boldsymbol{\Theta}(t)$ depends on $(\mathbf{X}, \mathbf{y})$ through GD updates
- Complex nonlinear dependence $\Rightarrow$ standard RMT does not work

Key insight: Although nonlinear, each GD update is **linear in X**

Proof sketch: Setup

(WLOG consider $m \gg 1$; the case $m = 1$ is similar)

Object of interest: Hessian diagonal block

$$\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X}^\top \mathbf{G}_j(t) \mathbf{X}$$

Intuition: y_i only depends on **few directions** $\boldsymbol{\Theta}_*^\top \mathbf{x}_i$
$\Rightarrow$ $\mathbf{G}_j(t)$ carries low-rank informative signal $\Rightarrow$ expect outliers

Why is this hard? Not a standard spiked random matrix

- $\boldsymbol{\Theta}(t)$ depends on $(\mathbf{X}, \mathbf{y})$ through GD updates
- Complex nonlinear dependence $\Rightarrow$ standard RMT does not work

Key insight: Although nonlinear, each GD update is **linear in X**

Proof sketch: Setup

(WLOG consider $m \gg 1$; the case $m = 1$ is similar)

Object of interest: Hessian diagonal block

$$\mathbf{H}_j(t) = \frac{1}{n} \sum_{i=1}^{n} g(y_i, \boldsymbol{\Theta}(t)^\top \mathbf{x}_i; j) \cdot \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X}^\top \mathbf{G}_j(t) \mathbf{X}$$

Intuition: y_i only depends on **few directions** $\boldsymbol{\Theta}_*^\top \mathbf{x}_i$
$\Rightarrow$ $\mathbf{G}_j(t)$ carries low-rank informative signal $\Rightarrow$ expect outliers

Why is this hard? Not a standard spiked random matrix

- $\boldsymbol{\Theta}(t)$ depends on $(\mathbf{X}, \mathbf{y})$ through GD updates
- Complex nonlinear dependence $\Rightarrow$ standard RMT does not work

Key insight: Although nonlinear, each GD update is **linear in X**

Proof sketch: Step 1 – Gaussian conditioning

Gaussian conditioning decomposition:

$$\mathbf{X} \overset{d}{=} \mathbf{P}_{\mathbf{F}}^{\perp} \mathbf{X}_{\text{new}} \mathbf{P}_{\boldsymbol{\Theta}}^{\perp} + \mathbf{X} \mathbf{P}_{\boldsymbol{\Theta}}$$

- $\mathbf{X}_{\text{new}}$: fresh i.i.d. $\mathsf{N}(0,1)$ matrix, **independent** of training data
- $\mathbf{P}_{\boldsymbol{\Theta}}$, $\mathbf{P}_{\mathbf{F}}$: projectors onto parameter/gradient subspaces

This technique has been extensively studied in AMP literature[8].

Result:

$$\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^{\top} \mathbf{G}_j(t) \mathbf{X}_{\text{new}} + \text{finite-rank perturbation}$$

$\Rightarrow$ Bulk follows **generalized Marchenko-Pastur law**
$\Rightarrow$ Outliers come from finite-rank perturbation

[8] [Bayati-Montanari '10, Donoho-Montanari '13]

Proof sketch: Step 1 – Gaussian conditioning

Gaussian conditioning decomposition:

$$\mathbf{X} \overset{d}{=} \mathbf{P}_{\mathbf{F}}^{\perp} \mathbf{X}_{\text{new}} \mathbf{P}_{\boldsymbol{\Theta}}^{\perp} + \mathbf{X} \mathbf{P}_{\boldsymbol{\Theta}}$$

- $\mathbf{X}_{\text{new}}$: fresh i.i.d. $\mathsf{N}(0,1)$ matrix, **independent** of training data
- $\mathbf{P}_{\boldsymbol{\Theta}}$, $\mathbf{P}_{\mathbf{F}}$: projectors onto parameter/gradient subspaces

This technique has been extensively studied in AMP literature[8].

Result:

$$\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^{\top} \mathbf{G}_j(t) \mathbf{X}_{\text{new}} + \text{finite-rank perturbation}$$

$\Rightarrow$ Bulk follows **generalized Marchenko-Pastur law**
$\Rightarrow$ Outliers come from finite-rank perturbation

[8] [Bayati-Montanari '10, Donoho-Montanari '13]

Proof sketch: Step 1 – Gaussian conditioning

Gaussian conditioning decomposition:

$$\mathbf{X} \overset{d}{=} \mathbf{P}_{\mathbf{F}}^{\perp} \mathbf{X}_{\text{new}} \mathbf{P}_{\boldsymbol{\Theta}}^{\perp} + \mathbf{X} \mathbf{P}_{\boldsymbol{\Theta}}$$

- $\mathbf{X}_{\text{new}}$: fresh i.i.d. $\mathsf{N}(0,1)$ matrix, **independent** of training data
- $\mathbf{P}_{\boldsymbol{\Theta}}$, $\mathbf{P}_{\mathbf{F}}$: projectors onto parameter/gradient subspaces

This technique has been extensively studied in AMP literature[8].

Result:

$$\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^{\top} \mathbf{G}_j(t) \mathbf{X}_{\text{new}} + \text{finite-rank perturbation}$$

$\Rightarrow$ Bulk follows **generalized Marchenko-Pastur law**
$\Rightarrow$ Outliers come from finite-rank perturbation

[8] [Bayati-Montanari '10, Donoho-Montanari '13]

Proof sketch: Step 1 – Gaussian conditioning

Gaussian conditioning decomposition:

$$\mathbf{X} \overset{d}{=} \mathbf{P}_{\mathbf{F}}^{\perp} \mathbf{X}_{\text{new}} \mathbf{P}_{\boldsymbol{\Theta}}^{\perp} + \mathbf{X} \mathbf{P}_{\boldsymbol{\Theta}}$$

- $\mathbf{X}_{\text{new}}$: fresh i.i.d. $\mathsf{N}(0,1)$ matrix, **independent** of training data
- $\mathbf{P}_{\boldsymbol{\Theta}}$, $\mathbf{P}_{\mathbf{F}}$: projectors onto parameter/gradient subspaces

This technique has been extensively studied in AMP literature[8].

Result:

$$\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^{\top} \mathbf{G}_j(t) \mathbf{X}_{\text{new}} + \text{finite-rank perturbation}$$

$\Rightarrow$ Bulk follows **generalized Marchenko-Pastur law**
$\Rightarrow$ Outliers come from finite-rank perturbation

[8] [Bayati-Montanari '10, Donoho-Montanari '13]

Proof sketch: Step 1 – Gaussian conditioning

Gaussian conditioning decomposition:

$$\mathbf{X} \overset{d}{=} \mathbf{P}_{\mathbf{F}}^{\perp} \mathbf{X}_{\text{new}} \mathbf{P}_{\boldsymbol{\Theta}}^{\perp} + \mathbf{X} \mathbf{P}_{\boldsymbol{\Theta}}$$

- $\mathbf{X}_{\text{new}}$: fresh i.i.d. $\mathsf{N}(0,1)$ matrix, **independent** of training data
- $\mathbf{P}_{\boldsymbol{\Theta}}$, $\mathbf{P}_{\mathbf{F}}$: projectors onto parameter/gradient subspaces

This technique has been extensively studied in AMP literature[8].

Result:

$$\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^{\top} \mathbf{G}_j(t) \mathbf{X}_{\text{new}} + \text{finite-rank perturbation}$$

$\Rightarrow$ Bulk follows **generalized Marchenko-Pastur law**
$\Rightarrow$ Outliers come from finite-rank perturbation

[8] [Bayati-Montanari '10, Donoho-Montanari '13]

Proof sketch: Step 2 – Outlier detection

From Step 1: $\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} + \text{finite-rank}$

Woodbury identity: For $A + UCV$,

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Apply to resolvent:

$$(\mathbf{H}_j(t) - zI)^{-1} = R_0(z) - R_0(z) \cdot [\text{low-rank}] \cdot \mathbf{M}_j(z;t)^{-1} \cdot [\text{low-rank}]^\top \cdot R_0(z)$$

where $R_0(z) = (\mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} - zI)^{-1}$ is the **bulk resolvent**

Key: z is an outlier eigenvalue $\Leftrightarrow \det(\mathbf{M}_j(z;t)) = 0$

$\mathbf{M}_j(z;t)$ is finite-dimensional (size $\sim k + mt$, independent of n, d)

Proof sketch: Step 2 – Outlier detection

From Step 1: $\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} + \text{finite-rank}$

Woodbury identity: For $A + UCV$,

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Apply to resolvent:

$$(\mathbf{H}_j(t) - zI)^{-1} = R_0(z) - R_0(z) \cdot [\text{low-rank}] \cdot \mathbf{M}_j(z;t)^{-1} \cdot [\text{low-rank}]^\top \cdot R_0(z)$$

where $R_0(z) = (\mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} - zI)^{-1}$ is the **bulk resolvent**

Key: z is an outlier eigenvalue $\Leftrightarrow \det(\mathbf{M}_j(z;t)) = 0$

$\mathbf{M}_j(z;t)$ is finite-dimensional (size $\sim k + mt$, independent of n, d)

Proof sketch: Step 2 – Outlier detection

From Step 1: $\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} + \text{finite-rank}$

Woodbury identity: For $A + UCV$,

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Apply to resolvent:

$$(\mathbf{H}_j(t) - zI)^{-1} = R_0(z) - R_0(z) \cdot [\text{low-rank}] \cdot \mathbf{M}_j(z;t)^{-1} \cdot [\text{low-rank}]^\top \cdot R_0(z)$$

where $R_0(z) = (\mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} - zI)^{-1}$ is the **bulk resolvent**

Key: z is an outlier eigenvalue $\Leftrightarrow \det(\mathbf{M}_j(z;t)) = 0$

$\mathbf{M}_j(z;t)$ is finite-dimensional (size $\sim k + mt$, independent of n, d)

Proof sketch: Step 2 – Outlier detection

From Step 1: $\mathbf{H}_j(t) \overset{d}{=} \mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} + \text{finite-rank}$

Woodbury identity: For $A + UCV$,

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Apply to resolvent:

$$(\mathbf{H}_j(t) - zI)^{-1} = R_0(z) - R_0(z) \cdot [\text{low-rank}] \cdot \mathbf{M}_j(z;t)^{-1} \cdot [\text{low-rank}]^\top \cdot R_0(z)$$

where $R_0(z) = (\mathbf{X}_{\text{new}}^\top \mathbf{G}_j \mathbf{X}_{\text{new}} - zI)^{-1}$ is the **bulk resolvent**

Key: z is an outlier eigenvalue $\Leftrightarrow \det(\mathbf{M}_j(z;t)) = 0$

$\mathbf{M}_j(z;t)$ is finite-dimensional (size $\sim k + mt$, independent of n, d)

Proof sketch: Step 2 – Block structure

Concentration: $\mathbf{M}_j(z;t) \xrightarrow{p} M_j^\infty(z;t)$ (deterministic limit)

Fortunately: $M_j^\infty(z;t)$ has **block-diagonal structure**!

$$\det(M_j^\infty(z;t)) = 0 \quad \Leftrightarrow \quad \det(M_{j,H}^\infty) = 0 \text{ or } \det(M_{j,R}^\infty) = 0$$

- $M_{j,H}^\infty$: **Hard block** (size $r \times r$, independent of t)
  $\Rightarrow \det(M_{j,H}^\infty) = 0$ gives our **outlier equation**
- $M_{j,R}^\infty$: **Rest block**
  $\Rightarrow$ Can prove: solutions have **no informative eigenvectors**

Proof sketch: Step 2 – Block structure

Concentration: $\mathbf{M}_j(z;t) \xrightarrow{p} M_j^\infty(z;t)$ (deterministic limit)

Fortunately: $M_j^\infty(z;t)$ has **block-diagonal structure**!

$$\det(M_j^\infty(z;t)) = 0 \quad \Leftrightarrow \quad \det(M_{j,H}^\infty) = 0 \text{ or } \det(M_{j,R}^\infty) = 0$$

- $M_{j,H}^\infty$: **Hard block** (size $r \times r$, independent of t)
  $\Rightarrow \det(M_{j,H}^\infty) = 0$ gives our **outlier equation**
- $M_{j,R}^\infty$: **Rest block**
  $\Rightarrow$ Can prove: solutions have **no informative eigenvectors**

Proof sketch: Step 2 – Block structure

Concentration: $\mathbf{M}_j(z;t) \xrightarrow{p} M_j^\infty(z;t)$ (deterministic limit)

Fortunately: $M_j^\infty(z;t)$ has **block-diagonal structure**!

$$\det(M_j^\infty(z;t)) = 0 \quad \Leftrightarrow \quad \det(M_{j,H}^\infty) = 0 \text{ or } \det(M_{j,R}^\infty) = 0$$

- $M_{j,H}^\infty$: **Hard block** (size $r \times r$, independent of t)
  $\Rightarrow \det(M_{j,H}^\infty) = 0$ gives our **outlier equation**
- $M_{j,R}^\infty$: **Rest block**
  $\Rightarrow$ Can prove: solutions have **no informative eigenvectors**

Proof sketch: Step 2 – Block structure

Concentration: $\mathbf{M}_j(z;t) \xrightarrow{p} M_j^\infty(z;t)$ (deterministic limit)

Fortunately: $M_j^\infty(z;t)$ has **block-diagonal structure**!

$$\det(M_j^\infty(z;t)) = 0 \quad \Leftrightarrow \quad \det(M_{j,H}^\infty) = 0 \text{ or } \det(M_{j,R}^\infty) = 0$$

- $M_{j,H}^\infty$: **Hard block** (size $r \times r$, independent of t)
  $\Rightarrow \det(M_{j,H}^\infty) = 0$ gives our **outlier equation**
- $M_{j,R}^\infty$: **Rest block**
  $\Rightarrow$ Can prove: solutions have **no informative eigenvectors**

Proof sketch: Step 3 – Eigenvector alignment

Residue formula:

$$\xi\xi^\top = -\frac{1}{2\pi i}\oint_\gamma (H_j(t) - zI)^{-1}\,dz$$

Correlation:

$$\|\Theta_{*H}^\top \xi\|^2 = -\frac{1}{2\pi i}\oint_\gamma \mathrm{Tr}\big(\Theta_{*H}^\top (H_j(t) - zI)^{-1}\Theta_{*H}\big)\,dz$$

Simplification:

- Substitute resolvent expansion from the last step
- $R_0(z)$ analytic inside $\gamma \Rightarrow$ only pole from $M_j(z;t)^{-1}$
- Replace $M_j(z;t)$ by limit $M_j^\infty(z;t)$, apply residue theorem
- Use block structure of $M_j^\infty \Rightarrow$ final formula

Proof sketch: Step 3 – Eigenvector alignment

Residue formula:

$$\xi\xi^\top = -\frac{1}{2\pi i}\oint_\gamma (H_j(t) - zI)^{-1}\,dz$$

Correlation:

$$\|\Theta_{*H}^\top \xi\|^2 = -\frac{1}{2\pi i}\oint_\gamma \mathrm{Tr}\big(\Theta_{*H}^\top (H_j(t) - zI)^{-1}\Theta_{*H}\big)\,dz$$

Simplification:

- Substitute resolvent expansion from the last step
- $R_0(z)$ analytic inside $\gamma \Rightarrow$ only pole from $M_j(z;t)^{-1}$
- Replace $M_j(z;t)$ by limit $M_j^\infty(z;t)$, apply residue theorem
- Use block structure of $M_j^\infty \Rightarrow$ final formula

Proof sketch: Step 3 – Eigenvector alignment

Residue formula:

$$\xi\xi^\top = -\frac{1}{2\pi i}\oint_\gamma (H_j(t) - zI)^{-1}\,dz$$

Correlation:

$$\|\Theta_{*H}^\top \xi\|^2 = -\frac{1}{2\pi i}\oint_\gamma \mathrm{Tr}\big(\Theta_{*H}^\top (H_j(t) - zI)^{-1}\Theta_{*H}\big)\,dz$$

Simplification:

- Substitute resolvent expansion from the last step
- $R_0(z)$ analytic inside $\gamma \Rightarrow$ only pole from $M_j(z;t)^{-1}$
- Replace $M_j(z;t)$ by limit $M_j^\infty(z;t)$, apply residue theorem
- Use block structure of $M_j^\infty \Rightarrow$ final formula

Proof sketch: Step 3 – Eigenvector alignment

Residue formula:

$$\xi\xi^\top = -\frac{1}{2\pi i}\oint_\gamma (\mathbf{H}_j(t) - zI)^{-1}\,dz$$

Correlation:

$$\|\mathbf{\Theta}_{*H}^\top \xi\|^2 = -\frac{1}{2\pi i}\oint_\gamma \mathrm{Tr}\big(\mathbf{\Theta}_{*H}^\top (\mathbf{H}_j(t) - zI)^{-1}\mathbf{\Theta}_{*H}\big)\,dz$$

Simplification:

- Substitute resolvent expansion from the last step
- $R_0(z)$ analytic inside $\gamma \Rightarrow$ only pole from $\mathbf{M}_j(z;t)^{-1}$
- Replace $\mathbf{M}_j(z;t)$ by limit $M_j^\infty(z;t)$, apply residue theorem
- Use block structure of $M_j^\infty \Rightarrow$ final formula

Proof sketch: Step 3 – Eigenvector alignment

Residue formula:

$$\xi\xi^\top = -\frac{1}{2\pi i}\oint_\gamma (\mathbf{H}_j(t) - zI)^{-1}\,dz$$

Correlation:

$$\|\mathbf{\Theta}_{*H}^\top \xi\|^2 = -\frac{1}{2\pi i}\oint_\gamma \mathrm{Tr}\big(\mathbf{\Theta}_{*H}^\top (\mathbf{H}_j(t) - zI)^{-1}\mathbf{\Theta}_{*H}\big)\,dz$$

Simplification:

- Substitute resolvent expansion from the last step
- $R_0(z)$ analytic inside $\gamma \Rightarrow$ only pole from $\mathbf{M}_j(z;t)^{-1}$
- Replace $\mathbf{M}_j(z;t)$ by limit $M_j^\infty(z;t)$, apply residue theorem
- Use block structure of $M_j^\infty \Rightarrow$ final formula

Proof sketch: Step 3 – Eigenvector alignment

Residue formula:

$$\xi\xi^\top = -\frac{1}{2\pi i}\oint_\gamma (\mathbf{H}_j(t) - zI)^{-1}\,dz$$

Correlation:

$$\|\mathbf{\Theta}_{*H}^\top \xi\|^2 = -\frac{1}{2\pi i}\oint_\gamma \mathrm{Tr}\big(\mathbf{\Theta}_{*H}^\top (\mathbf{H}_j(t) - zI)^{-1}\mathbf{\Theta}_{*H}\big)\,dz$$

Simplification:

- Substitute resolvent expansion from the last step
- $R_0(z)$ analytic inside $\gamma \Rightarrow$ only pole from $\mathbf{M}_j(z;t)^{-1}$
- Replace $\mathbf{M}_j(z;t)$ by limit $M_j^\infty(z;t)$, apply residue theorem
- Use block structure of $M_j^\infty \Rightarrow$ final formula