The merged-staircase property:

a necessary and nearly sufficient condition for SGD learning of sparse functions on two-layer neural networks

Emmanuel Abbe[1], **Enric Boix-Adsera**[2], **Theodor Misiakiewicz**[3]

[1]EPFL, [2]MIT, [3]Stanford

July 3rd, 2022

35th Annual Conference on Learning Theory (COLT 2022)

What functions can neural networks learn?

- NNs routinely solve **high-dimensional problems.**
- Learning general functions in HD is plagued by the **curse of dimensionality**.
- **Hypothesis:** NNs are good at learning functions with a **latent low-dimensional (sparse)** structure.
  - Approximation [Barron, 1993].
  - Generalization [Bach, 2017].
  - Computation: sparsity is not the right measure for computational complexity (above papers do not provide efficient algorithms). We expect some sparse functions to be easier to learn than others.
    In practice, NNs are trained with SGD.

Which sparse functions are efficiently learned by SGD-trained NNs?

What functions can neural networks learn?

- NNs routinely solve **high-dimensional problems.**
- Learning general functions in HD is plagued by the **curse of dimensionality**.
- **Hypothesis:** NNs are good at learning functions with a **latent low-dimensional (sparse)** structure.
  - Approximation [Barron, 1993].
  - Generalization [Bach, 2017].
  - Computation: sparsity is not the right measure for computational complexity (above papers do not provide efficient algorithms). We expect some sparse functions to be easier to learn than others.
    In practice, NNs are trained with SGD.

Which sparse functions are efficiently learned by SGD-trained NNs?

What functions can neural networks learn?

- NNs routinely solve **high-dimensional problems.**
- Learning general functions in HD is plagued by the **curse of dimensionality**.
- **Hypothesis:** NNs are good at learning functions with a **latent low-dimensional (sparse)** structure.
  - Approximation [Barron, 1993].
  - Generalization [Bach, 2017].
  - Computation: sparsity is not the right measure for computational complexity (above papers do not provide efficient algorithms). We expect some sparse functions to be easier to learn than others.
    In practice, NNs are trained with SGD.

Which sparse functions are efficiently learned by SGD-trained NNs?

What functions can neural networks learn?

- NNs routinely solve **high-dimensional problems.**
- Learning general functions in HD is plagued by the **curse of dimensionality**.
- **Hypothesis:** NNs are good at learning functions with a **latent low-dimensional (sparse)** structure.
  - Approximation [Barron, 1993].
  - Generalization [Bach, 2017].
  - Computation: sparsity is not the right measure for computational complexity (above papers do not provide efficient algorithms). We expect some sparse functions to be easier to learn than others.
    In practice, NNs are trained with SGD.

Which sparse functions are efficiently learned by SGD-trained NNs?

Learning sparse functions on the hypercube

- Consider $\boldsymbol{x} \sim \mathrm{Unif}(\{+1,-1\}^d)$ and target function

$$f_*(\boldsymbol{x}) = h_*(\boldsymbol{z}), \quad \boldsymbol{z} \in \{+1,-1\}^P \text{ latent (unknown) support } (P \ll d).$$

- Examples:

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3\,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3\,.$$

  Are these functions equivalent for SGD-trained NNs?

- In the **"lazy" regime** (NTK regime), no adaptation to sparsity:

Proposition [Abbe,Boix-Adsera,Misiakiewicz]

Any linear method (NTK, kernel methods, random feature models...) require $n = \Omega_d(d^P)$ samples to learn any degree-P polynomial h_*.

- What about SGD in the **feature learning regime**?

Learning sparse functions on the hypercube

- Consider $\boldsymbol{x} \sim \mathrm{Unif}(\{+1,-1\}^d)$ and target function

$$f_*(\boldsymbol{x}) = h_*(\boldsymbol{z}), \quad \boldsymbol{z} \in \{+1,-1\}^P \text{ latent (unknown) support } (P \ll d).$$

- Examples:

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3\,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3\,.$$

  Are these functions equivalent for SGD-trained NNs?

- In the **"lazy" regime** (NTK regime), no adaptation to sparsity:

Proposition [Abbe,Boix-Adsera,Misiakiewicz]

Any linear method (NTK, kernel methods, random feature models...) require $n = \Omega_d(d^P)$ samples to learn any degree-P polynomial h_*.

- What about SGD in the **feature learning regime**?

Learning sparse functions on the hypercube

- Consider $\boldsymbol{x} \sim \mathrm{Unif}(\{+1,-1\}^d)$ and target function

$$f_*(\boldsymbol{x}) = h_*(\boldsymbol{z}), \quad \boldsymbol{z} \in \{+1,-1\}^P \text{ latent (unknown) support } (P \ll d).$$

- Examples:

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3\,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3\,.$$

  Are these functions equivalent for SGD-trained NNs?

- In the **"lazy" regime** (NTK regime), no adaptation to sparsity:

Proposition [Abbe,Boix-Adsera,Misiakiewicz]

Any linear method (NTK, kernel methods, random feature models...) require $n = \Omega_d(d^P)$ samples to learn any degree-P polynomial h_*.

- What about SGD in the **feature learning regime**?

Learning sparse functions on the hypercube

- Consider $\boldsymbol{x} \sim \mathrm{Unif}(\{+1,-1\}^d)$ and target function

$$f_*(\boldsymbol{x}) = h_*(\boldsymbol{z}), \quad \boldsymbol{z} \in \{+1,-1\}^P \text{ latent (unknown) support } (P \ll d).$$

- Examples:

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3\,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3\,.$$

  Are these functions equivalent for SGD-trained NNs?

- In the **"lazy" regime** (NTK regime), no adaptation to sparsity:

Proposition [Abbe,Boix-Adsera,Misiakiewicz]

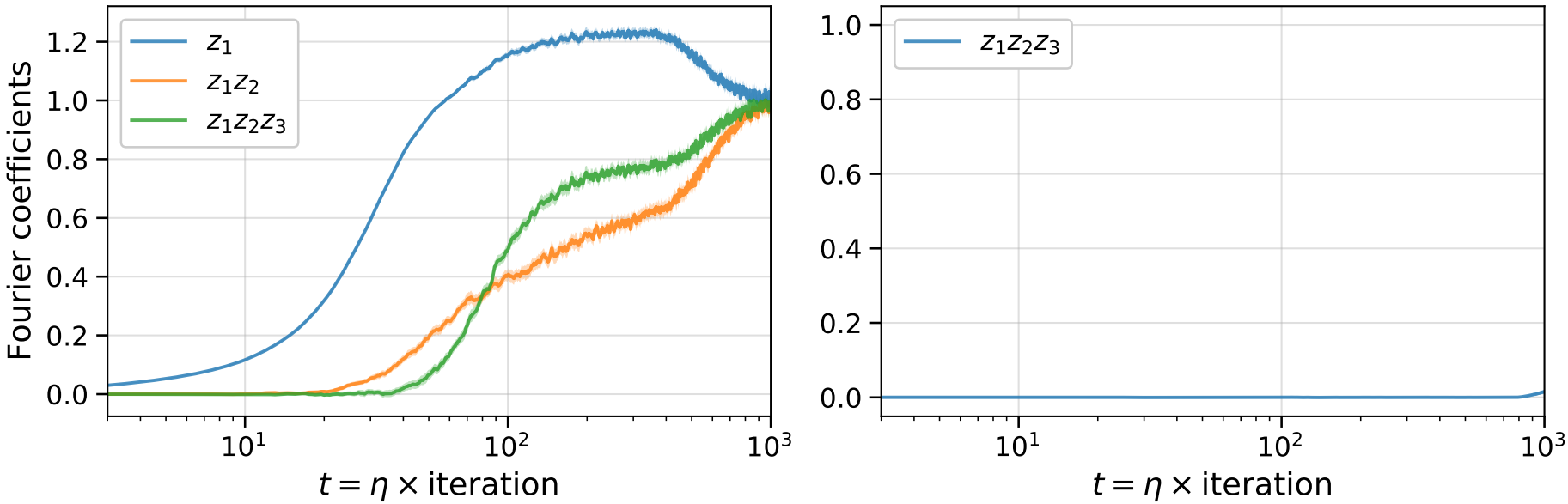
Any linear method (NTK, kernel methods, random feature models...) require $n = \Omega_d(d^P)$ samples to learn any degree-P polynomial h_*.

- What about SGD in the **feature learning regime**?

NNs trained in the mean-field regime

- Take **2-layer neural network** with M neurons in the **mean-field scaling** $\Theta(1/M)$.
- Train with vanilla online SGD with step size $\eta = \Theta(1/d)$.
- Simulations with $d = M = 300$:

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3 \,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3 \,.$$

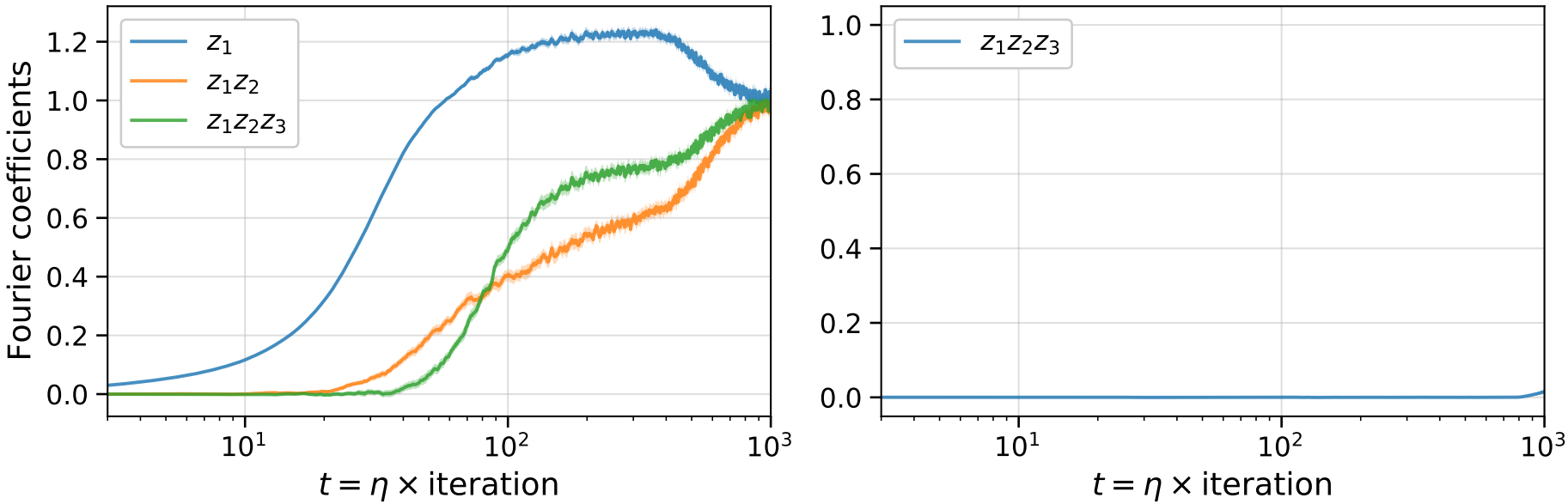


Can we characterize which sparse functions are learnable by SGD in $O(d)$ steps?

NNs trained in the mean-field regime

- Take **2-layer neural network** with M neurons in the **mean-field scaling** $\Theta(1/M)$.
- Train with vanilla online SGD with step size $\eta = \Theta(1/d)$.
- Simulations with $d = M = 300$:

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3 \,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3 \,.$$

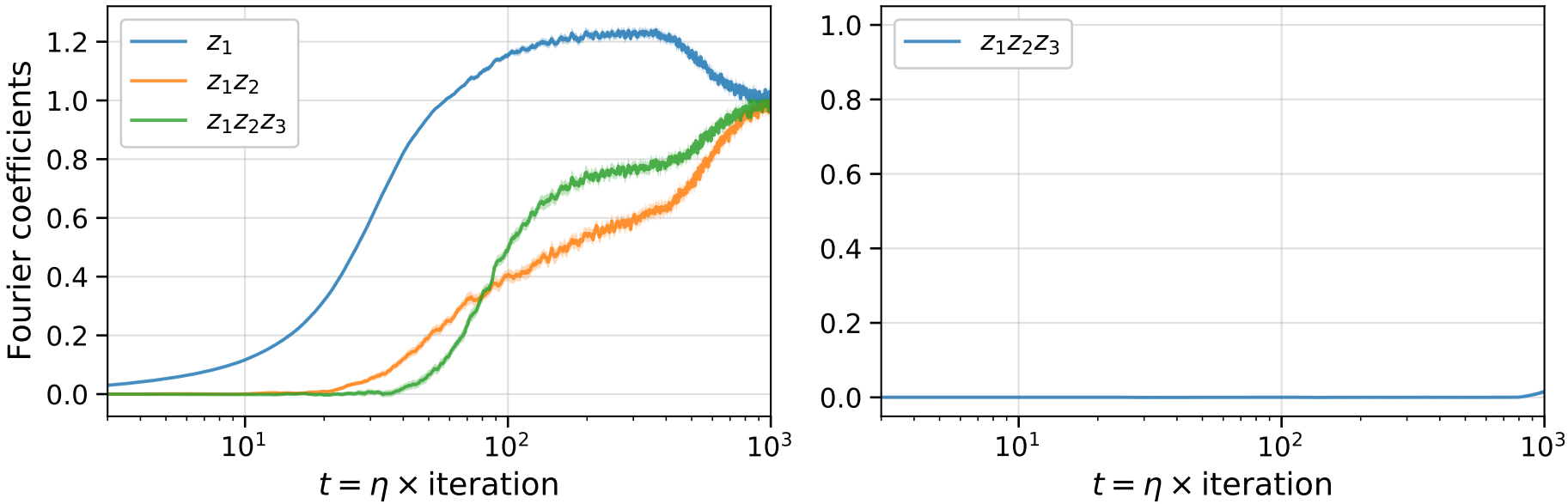


Can we characterize which sparse functions are learnable by SGD in $O(d)$ steps?

NNs trained in the mean-field regime

- Take **2-layer neural network** with M neurons in the **mean-field scaling** $\Theta(1/M)$.
- Train with vanilla online SGD with step size $\eta = \Theta(1/d)$.
- Simulations with $d = M = 300$:

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3 \,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3 \,.$$



Can we characterize which sparse functions are learnable by SGD in $O(d)$ steps?

Merged staircase functions

Fourier coefficient for $S \subseteq [P]$: $\hat{h}_*(S) = \mathbb{E}_{\boldsymbol{z}}\Big[h_*(\boldsymbol{z})\chi_S(\boldsymbol{z})\Big]$ where $\chi_S(\boldsymbol{z}) = \prod_{i\in S} z_i$.

$$h_*(\boldsymbol{z}) = \sum_{S\in\mathcal{Q}} \hat{h}_*(S)\chi_S(\boldsymbol{z})\,,$$

where $\mathcal{Q}$ contains all non-zero Fourier coefficients $\hat{h}_*(S) \neq 0$.

Merged-Staircase property (MSP)

$h_* : \{-1,+1\}^P \to \mathbb{R}$ has the *merged-staircase property* (MSP) if we can write elements of $\mathcal{Q}$ in order $(S_1, \ldots, S_r)$ such that for any $j \in [r]$, we have $|S_j \setminus (S_1 \cup \ldots \cup S_{j-1})| \leq 1$.

Examples of MSP functions:

$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_1z_2z_3 + z_1z_2z_3z_4\,,$$
$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_2z_3 + z_3z_4 + z_3z_4z_5\,.$$

Examples of non-MSP functions:

$$h_*(\boldsymbol{z}) = z_1 + z_1z_2z_3 + z_1z_2z_3z_4\,,$$
$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_3z_4 + z_3z_4z_5\,.$$

Merged staircase functions

Fourier coefficient for $S \subseteq [P]$: $\hat{h}_*(S) = \mathbb{E}_{\boldsymbol{z}}\Big[h_*(\boldsymbol{z})\chi_S(\boldsymbol{z})\Big]$ where $\chi_S(\boldsymbol{z}) = \prod_{i\in S} z_i$.

$$h_*(\boldsymbol{z}) = \sum_{S\in\mathcal{Q}} \hat{h}_*(S)\chi_S(\boldsymbol{z})\,,$$

where $\mathcal{Q}$ contains all non-zero Fourier coefficients $\hat{h}_*(S) \neq 0$.

Merged-Staircase property (MSP)

$h_* : \{-1,+1\}^P \to \mathbb{R}$ has the *merged-staircase property* (MSP) if we can write elements of $\mathcal{Q}$ in order $(S_1, \ldots, S_r)$ such that for any $j \in [r]$, we have $|S_j \setminus (S_1 \cup \ldots \cup S_{j-1})| \leq 1$.

Examples of MSP functions:

$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_1z_2z_3 + z_1z_2z_3z_4\,,$$
$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_2z_3 + z_3z_4 + z_3z_4z_5\,.$$

Examples of non-MSP functions:

$$h_*(\boldsymbol{z}) = z_1 + z_1z_2z_3 + z_1z_2z_3z_4\,,$$
$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_3z_4 + z_3z_4z_5\,.$$

Merged staircase functions

Fourier coefficient for $S \subseteq [P]$: $\hat{h}_*(S) = \mathbb{E}_{\boldsymbol{z}}\Big[h_*(\boldsymbol{z})\chi_S(\boldsymbol{z})\Big]$ where $\chi_S(\boldsymbol{z}) = \prod_{i\in S} z_i$.

$$h_*(\boldsymbol{z}) = \sum_{S\in\mathcal{Q}} \hat{h}_*(S)\chi_S(\boldsymbol{z})\,,$$

where $\mathcal{Q}$ contains all non-zero Fourier coefficients $\hat{h}_*(S) \neq 0$.

Merged-Staircase property (MSP)

$h_* : \{-1,+1\}^P \to \mathbb{R}$ has the *merged-staircase property* (MSP) if we can write elements of $\mathcal{Q}$ in order $(S_1, \ldots, S_r)$ such that for any $j \in [r]$, we have $|S_j \setminus (S_1 \cup \ldots \cup S_{j-1})| \leq 1$.

Examples of MSP functions:

$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_1z_2z_3 + z_1z_2z_3z_4\,,$$
$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_2z_3 + z_3z_4 + z_3z_4z_5\,.$$

Examples of non-MSP functions:

$$h_*(\boldsymbol{z}) = z_1 + z_1z_2z_3 + z_1z_2z_3z_4\,,$$
$$h_*(\boldsymbol{z}) = z_1 + z_1z_2 + z_3z_4 + z_3z_4z_5\,.$$

Main result

Theorem 1 [Abbe,Boix-Adsera,Misiakiewicz]

MSP is necessary and nearly sufficient* to learn h_* in $O(d)$ steps/samples of online SGD** in mean-field regime.

*Excludes a set of MSP fcts $h_*(\boldsymbol{z}) = \sum_{S\in\mathcal{Q}} h_*(S)\chi_S(\boldsymbol{z})$ with $\{h_*(S)\}_{S\in\mathcal{Q}}$ of measure 0. (This is unavoidable: some degenerate cases of MSP functions that are not learned)

**For the sufficiency result, we train the first layer, then the second layer

Example

$$\underbrace{h_{*,1}(\boldsymbol{z}) = z_1 + z_1z_2 + z_1z_2z_3}_{k=O_d(d) \text{ online SGD steps is enough}}, \qquad \underbrace{h_{*,2}(\boldsymbol{z}) = z_1z_2z_3}_{\text{needs } k \gg d \text{ steps}}.$$

Conjecture

For leap-ℓ MSP, $k = O_d(d\log(d))$ steps for $\ell = 2$ and $k = \tilde{O}_d(d^{\ell-1})$ for $\ell > 2$.

Main result

Theorem 1 [Abbe,Boix-Adsera,Misiakiewicz]

MSP is necessary and nearly sufficient* to learn h_* in $O(d)$ steps/samples of online SGD** in mean-field regime.

*Excludes a set of MSP fcts $h_*(\boldsymbol{z}) = \sum_{S\in\mathcal{Q}} h_*(S)\chi_S(\boldsymbol{z})$ with $\{h_*(S)\}_{S\in\mathcal{Q}}$ of measure 0. (This is unavoidable: some degenerate cases of MSP functions that are not learned)

**For the sufficiency result, we train the first layer, then the second layer

Example

$$\underbrace{h_{*,1}(\boldsymbol{z}) = z_1 + z_1z_2 + z_1z_2z_3}_{k=O_d(d) \text{ online SGD steps is enough}}, \qquad \underbrace{h_{*,2}(\boldsymbol{z}) = z_1z_2z_3}_{\text{needs } k \gg d \text{ steps}}.$$

Conjecture

For leap-ℓ MSP, $k = O_d(d\log(d))$ steps for $\ell = 2$ and $k = \tilde{O}_d(d^{\ell-1})$ for $\ell > 2$.

Main result

Theorem 1 [Abbe,Boix-Adsera,Misiakiewicz]

MSP is necessary and nearly sufficient* to learn h_* in $O(d)$ steps/samples of online SGD** in mean-field regime.

*Excludes a set of MSP fcts $h_*(\boldsymbol{z}) = \sum_{S\in\mathcal{Q}} h_*(S)\chi_S(\boldsymbol{z})$ with $\{h_*(S)\}_{S\in\mathcal{Q}}$ of measure 0. (This is unavoidable: some degenerate cases of MSP functions that are not learned)

**For the sufficiency result, we train the first layer, then the second layer

Example

$$\underbrace{h_{*,1}(\boldsymbol{z}) = z_1 + z_1z_2 + z_1z_2z_3}_{k=O_d(d) \text{ online SGD steps is enough}}, \qquad \underbrace{h_{*,2}(\boldsymbol{z}) = z_1z_2z_3}_{\text{needs } k \gg d \text{ steps}}.$$

Conjecture

For leap-ℓ MSP, $k = O_d(d\log(d))$ steps for $\ell = 2$ and $k = \tilde{O}_d(d^{\ell-1})$ for $\ell > 2$.

New technical tool: dimension-free dynamics

Previous work: SGD trajectory converges to **"mean-field dynamics"** (MF-PDE) in limit of $M \to \infty$ (large width), and $\eta \to 0$ (large sample size).

[Chizat, Bach,'18], [Mei, Montanari, Nguyen,'18] [Rotskoff,Vanden-Eijnden,'18], [Sirignano,Spiliopoulos,'18]

MF-PDE is Wasserstein gradient flow on $(d+1)$ dimensions.
Difficult to analyze!

Our work: Target function $f_*(x) = h_*(z)$ only depends on P input coordinates.

We approximate SGD trajectory by **"dimension-free mean-field dynamics"** (DF-PDE).

DF-PDE is Wasserstein gradient flow on $(P+2)$ dimensions.
More tractable since P is constant.

Theorem 2 [Abbe,Boix-Adsera,Misiakiewicz]

h_* is learnable in $O(d)$ steps in mean-field regime iff DF-PDE achieves arbitrarily small test error in $O(1)$ time.

New technical tool: dimension-free dynamics

Previous work: SGD trajectory converges to **"mean-field dynamics"** (MF-PDE) in limit of $M \to \infty$ (large width), and $\eta \to 0$ (large sample size).

[Chizat, Bach,'18], [Mei, Montanari, Nguyen,'18] [Rotskoff,Vanden-Eijnden,'18], [Sirignano,Spiliopoulos,'18]

MF-PDE is Wasserstein gradient flow on $(d+1)$ dimensions.
Difficult to analyze!

Our work: Target function $f_*(\boldsymbol{x}) = h_*(\boldsymbol{z})$ only depends on P input coordinates.

We approximate SGD trajectory by **"dimension-free mean-field dynamics"** (DF-PDE).

DF-PDE is Wasserstein gradient flow on $(P+2)$ dimensions.
More tractable since P is constant.

Theorem 2 [Abbe,Boix-Adsera,Misiakiewicz]

h_* is learnable in $O(d)$ steps in mean-field regime iff DF-PDE achieves arbitrarily small test error in $O(1)$ time.

New technical tool: dimension-free dynamics

Previous work: SGD trajectory converges to **"mean-field dynamics"** (MF-PDE) in limit of $M \to \infty$ (large width), and $\eta \to 0$ (large sample size).

[Chizat, Bach,'18], [Mei, Montanari, Nguyen,'18] [Rotskoff,Vanden-Eijnden,'18], [Sirignano,Spiliopoulos,'18]

MF-PDE is Wasserstein gradient flow on $(d+1)$ dimensions.
Difficult to analyze!

Our work: Target function $f_*(\boldsymbol{x}) = h_*(\boldsymbol{z})$ only depends on P input coordinates.

We approximate SGD trajectory by **"dimension-free mean-field dynamics"** (DF-PDE).

DF-PDE is Wasserstein gradient flow on $(P+2)$ dimensions.
More tractable since P is constant.

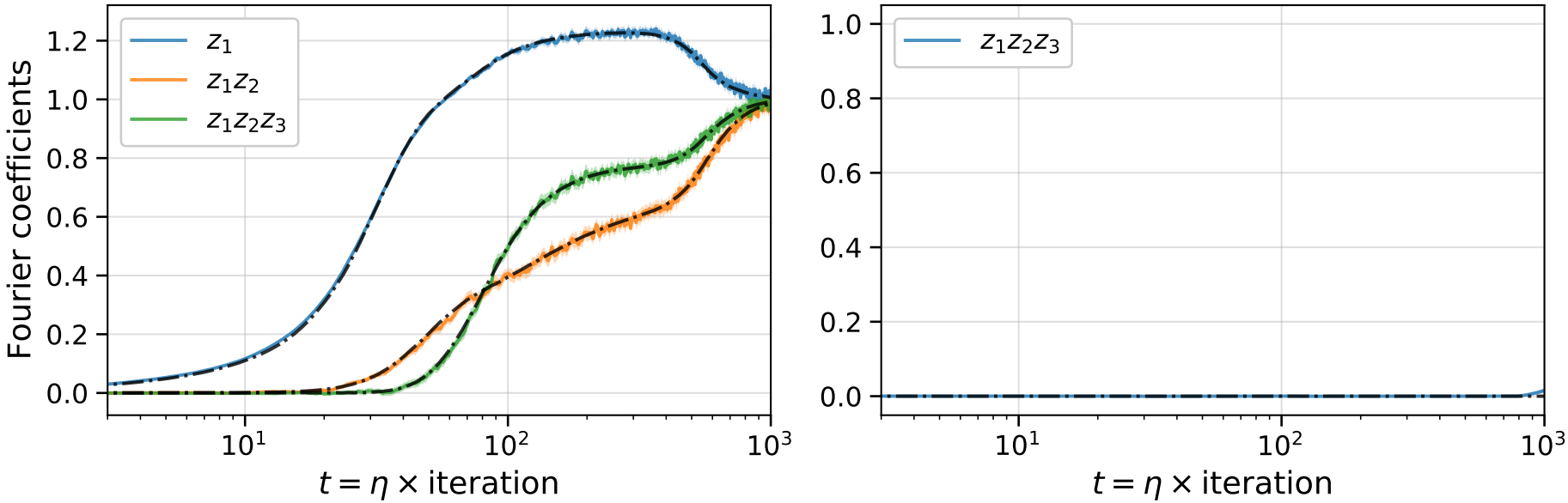
Theorem 2 [Abbe,Boix-Adsera,Misiakiewicz]

h_* is learnable in $O(d)$ steps in mean-field regime iff DF-PDE achieves arbitrarily small test error in $O(1)$ time.

Numerical simulation of DF-PDE

Take $d = 300$ and train with SGD on a 2-layer NN with $M = 300$

$$h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3 \,, \qquad h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3 \,.$$



Dashed line is DF-PDE prediction.

Intuition

- **Necessity result**:
  For non-MSP functions, like $h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3$, DF-PDE gets stuck in a saddle point.

- **Sufficiency result**:
  For MSP functions, like $h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3$, the low-degree terms allow escaping the saddle point.

  Study layerwise training to make it tractable to analyze:
  1. Train weights of layer 1 for a small time $T_1 = O(1)$.
  2. Train weights of layer 2 for time $T_2 = O(\log(1/\varepsilon))$.

  Proof ideas: Need to show kernel of layer 2 features is nonsingular after training of layer 1. To show, Taylor-expand the DF-PDE, and use polynomial anti-concentration.

Intuition

- **Necessity result**:
  For non-MSP functions, like $h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3$, DF-PDE gets stuck in a saddle point.

- **Sufficiency result**:
  For MSP functions, like $h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3$, the low-degree terms allow escaping the saddle point.

  Study layerwise training to make it tractable to analyze:
  1. Train weights of layer 1 for a small time $T_1 = O(1)$.
  2. Train weights of layer 2 for time $T_2 = O(\log(1/\varepsilon))$.

  Proof ideas: Need to show kernel of layer 2 features is nonsingular after training of layer 1. To show, Taylor-expand the DF-PDE, and use polynomial anti-concentration.

Conclusion and future directions

Summary

- Studied learnability in mean-field neural networks with $O(d)$ samples of online SGD.
- For sparse functions, **merged staircase property** is necessary and nearly sufficient for learnability in this regime.

Open problems:

- Beyond $O(d)$ samples: *conjecture* leap-ℓ MSP captures complexity for $O(d^\alpha)$ samples.
- Beyond the binary hypercube: extension to more realistic data distributions.

Thank you!