Learning latent low-dimensional functions with neural networks

Theodor Misiakiewicz

Stanford University

Joint work with Emmanuel Abbe (EPFL) and Enric Boix-Adsera (MIT)

April 24th, 2023

Workshop on Optimal Transport, Mean-Field Models, and Machine Learning,

TUM Institute for Advanced Study (Munich)

Curse of dimensionality

Two observations:

- Deep Learning routinely solve high-dimensional problems.
- **Curse of dimensionality:** $\mathcal{H}$ = 1-bounded 1-Lipschitz functions on $[0,1]^d$,

  With M neurons [Maiorov,'99] and n samples

$$\text{Approx. error} \asymp M^{-\Theta(1/d)}, \qquad \text{Gen. error} \asymp n^{-\Theta(1/d)}.$$

Why does DL seemingly avoid the curse of dimensionality?

Conjecture:

Real data has low-dimensional structure. NNs can adapt to it and break the CoD.

Simplest example: latent low-dimensional ("multi-index") functions, i.e., that depend on a latent (unknown) low-dimensional subspace.

There exist P directions $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_P)$ with $P \ll d$ such that

$$f_*(\boldsymbol{x}) = h_*(\langle \boldsymbol{u}_1, \boldsymbol{x} \rangle, \ldots, \langle \boldsymbol{u}_P, \boldsymbol{x} \rangle) \, .$$

- $\mathcal{H}_P = \{$functions in $\mathcal{H}$ that depend on P-coordinates$\}$

  [Bach,'17], [Schmidt-Hieber,'20], etc...

$$\text{Approx. error} \asymp M^{-\Theta(1/P)}, \qquad \text{Gen. error} \asymp n^{-\Theta(1/P)}.$$

Intuition: ERM with $M = \infty$ + sparsity inducing norm, $\sigma(\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle)$ with $\boldsymbol{w}_j$ aligned with the P-dimensional support.

NNs can break the CoD on multi-index fcts in approx./gen.

- However, these results do not provide efficient algorithms (only hold with unbounded computational resources).

  This is **unavoidable**, because of computational hardness results.
  [Klivans, Sherstov, '09], [Neyshabur, Tomokia, Srerbro, '15]

We expect some multi-index functions to be easier/harder to learn, which will not be captured by studying approximation and generalization alone.

Goal of this talk

Which multi-index functions are efficiently learned by NNs trained using SGD?

- Need to study the SGD training dynamics.
- Understand how SGD *dynamically* picks up the low-dimensional support.

Setting

- **Toy data distribution:** $\boldsymbol{x} \sim \mathrm{Unif}(\{+1,-1\}^d)$ and sparse target function

$$f_*(\boldsymbol{x}) = h_*(\boldsymbol{z}), \qquad \boldsymbol{z} \in \{\pm 1\}^P \text{ unknown subset of } P \text{ coordinates of } \boldsymbol{x},\ P \ll d.$$

- 2-layer neural network with M neurons:

$$\hat{f}_{\mathsf{NN}}(\boldsymbol{x};\boldsymbol{\Theta}) = \frac{1}{M}\sum_{j\in[M]} a_j \sigma(\langle \boldsymbol{w}_j, \boldsymbol{x}\rangle), \qquad \boldsymbol{\Theta} = (\boldsymbol{\theta}_j)_{j\in[M]} = (a_j, \boldsymbol{w}_j)_{j\in[M]}.$$

- **Goal:** fit the target function f_* by minimizing

$$\min_{\boldsymbol{\Theta}} R(f_*, \boldsymbol{\Theta}) = \mathbb{E}_{\boldsymbol{x}}\Big[\big(f_*(\boldsymbol{x}) - \hat{f}_{\mathsf{NN}}(\boldsymbol{x};\boldsymbol{\Theta})\big)^2\Big].$$

- **Online (one-pass) SGD:** initialization $(a_j, \boldsymbol{w}_j)_{j\in[M]} \sim_{\text{iid}} \rho_0$.
  Update: at each step k, fresh sample $(\boldsymbol{x}_k, y_k)$ with $y_k = f_*(\boldsymbol{x}_k) + \varepsilon_k$,

$$\boldsymbol{\theta}_j^{k+1} = \boldsymbol{\theta}_j^k + \eta\big(y_k - \hat{f}_{\mathsf{NN}}(\boldsymbol{x}_k;\boldsymbol{\Theta}^k)\big)\cdot \nabla_{\boldsymbol{\theta}_j}\big\{a_j\sigma(\langle \boldsymbol{x}_k, \boldsymbol{w}_j^k\rangle)\big\}.$$

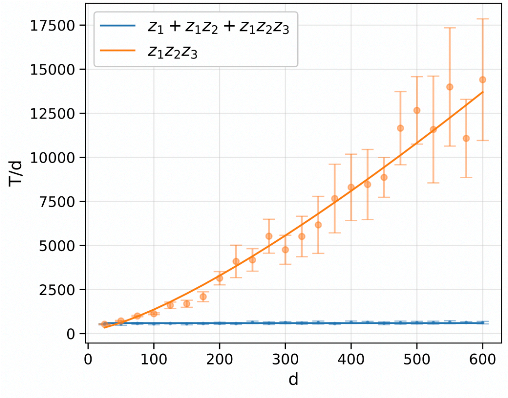
  (sample complexity n = number of SGD steps T)

Motivating examples

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

Are these 2 functions equivalent for SGD-trained NNs? If not, which one is easier to learn?

T = number of SGD steps to reach 0.05 test error.



$$\underbrace{h_{*,1}(z) = z_1 + z_1z_2 + z_1z_2z_3}_{T=n=\Theta(d) \text{ SGD steps to learn}}, \qquad \underbrace{h_{*,2}(z) = z_1z_2z_3}_{\text{needs } T = n = \widetilde{\Theta}(d^2) \text{ steps}} .$$

1. Which functions are learned in $\Theta(d)$ SGD steps?

We need to study the dynamics:

$$\boldsymbol{\theta}_j^{k+1} = \boldsymbol{\theta}_j^k + \eta\Big(y_k - \hat{f}_{\mathsf{NN}}(\boldsymbol{x}_k; \boldsymbol{\Theta}^k)\Big) \cdot \nabla_{\boldsymbol{\theta}_j} \Big\{ a_j \sigma(\langle \boldsymbol{x}_k, \boldsymbol{w}_j^k \rangle) \Big\} \,,$$

$$\hat{f}_{\mathsf{NN}}(\boldsymbol{x}; \boldsymbol{\Theta}^k) = \frac{1}{M} \sum_{j \in [M]} a_j^k \sigma(\langle \boldsymbol{w}_j^k, \boldsymbol{x} \rangle) \,, \qquad \boldsymbol{\Theta}^k = (\boldsymbol{\theta}_j^k)_{j \in [M]} = (a_j^k, \boldsymbol{w}_j^k)_{j \in [M]} \,.$$

Two approximations:

1. Mean-field approximation $M \to \infty$, $\eta \to 0$.
2. Ambient dimension $d \to \infty$.

1 Mean-field approximation

[Mei et al,'18], [Chizat,Bach,'18], [Rotskoff,Vanden-Eijnden,'18], [Sirignano,Spiliopoulos,'18]

- $M \to \infty$ limit: $(\boldsymbol{\theta}_j)_{j\in[M]}$ replaced by $\rho \in \mathcal{P}(\mathbb{R}^{d+1})$

$$\hat{f}_{\mathsf{NN}}(\boldsymbol{x};\boldsymbol{\Theta}) = \frac{1}{M}\sum_{j\in[M]} a_j\sigma(\langle \boldsymbol{w}_j,\boldsymbol{x}\rangle)\,, \quad \longrightarrow \quad \hat{f}_{\mathsf{NN}}(\boldsymbol{x};\rho) = \int a\sigma(\langle \boldsymbol{w},\boldsymbol{x}\rangle)\rho(\mathrm{d}\boldsymbol{\theta})\,.$$

- $\eta \to 0$ limit: gradient flow on the population loss, $(\rho_t)_{t\geq 0}$ solution of PDE with:

$$\boldsymbol{\theta}^t \sim \rho_t\,, \qquad \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\theta}^t = \mathbb{E}_{\boldsymbol{x}}\Big[\big(f_*(\boldsymbol{x}) - \hat{f}_{\mathsf{NN}}(\boldsymbol{x};\rho_t)\big)\nabla_{\boldsymbol{\theta}}\{a^t\sigma(\langle \boldsymbol{w}^t,\boldsymbol{x}\rangle)\}\Big]\,.$$

MF dynamics = gradient flow on population loss with $M = \infty$.

- [Mei, **M.**, Montanari,'19] with probability at least $1 - 1/M$:

$$\sup_{k\in\{0,\ldots,T\}} \left\| \hat{f}_{\mathsf{NN}}(\cdot;\boldsymbol{\Theta}^k) - \hat{f}_{\mathsf{NN}}(\cdot;\rho_{k\eta})\right\|_{L^2} \leq K e^{K(\eta T)^3}\Bigg[\underbrace{\sqrt{\frac{\log(M)}{M}}}_{M\to\infty} + \underbrace{\sqrt{d\eta}}_{\eta\to 0}\Bigg]\,.$$

2 Ambient dimension $d \to \infty$

Use the symmetry of the problem to show that MF dynamics is well approximated by a low-dim dynamics as $d \to \infty$ (with h_*, P fixed).

- $\boldsymbol{x} \sim \text{Unif}(\{+1, -1\}^d)$ and $\boldsymbol{x} = (\boldsymbol{z}, \boldsymbol{r})$, $\boldsymbol{z} \in \mathbb{R}^P$, $\boldsymbol{r} \in \mathbb{R}^{d-P}$, $f_*(\boldsymbol{x}) = h_*(\boldsymbol{z})$,

  First layer weights: $\boldsymbol{w}^t = (\boldsymbol{u}^t, \boldsymbol{v}^t)$, $\boldsymbol{u}^t \in \mathbb{R}^P$ and $\boldsymbol{v}^t \in \mathbb{R}^{d-P}$.

  For $\boldsymbol{w}^0 \sim \mathsf{N}(0, \kappa^2 \mathbf{I}_d / d)$:

$$\hat{f}_{\mathsf{NN}}(\boldsymbol{x}; \rho_0) = \int a^0 \sigma(\langle \boldsymbol{u}^0, \boldsymbol{z} \rangle + \langle \boldsymbol{v}^0, \boldsymbol{r} \rangle) \rho_0(\mathrm{d}\boldsymbol{\theta}^t) = \int a^0 \sigma(\langle \boldsymbol{u}^0, \boldsymbol{z} \rangle + \langle \boldsymbol{v}^0, \mathbf{1} \rangle) \rho_0(\mathrm{d}\boldsymbol{\theta}^t)$$

$$\Longrightarrow \quad \hat{f}_{\mathsf{NN}}(\boldsymbol{z}; \rho_t) = \mathbb{E}_{\boldsymbol{r}}[\hat{f}_{\mathsf{NN}}(\boldsymbol{x}; \rho_t)] = \int a^t \mathbb{E}_{\boldsymbol{r}}[\sigma(\langle \boldsymbol{u}^t, \boldsymbol{z} \rangle + \langle \boldsymbol{v}^t, \boldsymbol{r} \rangle)] \rho_t(\mathrm{d}\boldsymbol{\theta}^t) .$$

- As $d \to \infty$,

$$\mathbb{E}_{\boldsymbol{r}}[\sigma(\langle \boldsymbol{u}^t, \boldsymbol{z} \rangle + \langle \boldsymbol{v}^t, \boldsymbol{r} \rangle)] \to \mathbb{E}_G[\sigma(\langle \boldsymbol{u}^t, \boldsymbol{z} \rangle + \|\boldsymbol{v}^t\|_2 G)] =: \sigma_{\|\boldsymbol{v}^t\|_2}(\langle \boldsymbol{u}^t, \boldsymbol{z} \rangle) ,$$

  and $\boldsymbol{u}^0 \to 0$, $\|\boldsymbol{v}^0\| \to \kappa$.

Dimension-free dynamics

- As $d \to \infty$, $(\boldsymbol{a}^t, \boldsymbol{u}^t, \boldsymbol{v}^t) \sim \rho_t$ approximated by $\overline{\boldsymbol{\theta}}^t := (\overline{a}^t, \overline{\boldsymbol{u}}^t, \overline{s}^t) \sim \overline{\rho}_t \in \mathcal{P}(\mathbb{R}^{P+2})$
  - $\overline{\rho}_t$ follows a dimension free dynamics (DF-PDE):

$$\overline{\boldsymbol{\theta}}^t \sim \overline{\rho}_t \,, \qquad \frac{\mathrm{d}}{\mathrm{d}t}\overline{\boldsymbol{\theta}}^t = \mathbb{E}_{\boldsymbol{z}}\Big[\big(h_*(\boldsymbol{x}) - \hat{f}_{\mathsf{NN}}(\boldsymbol{z};\overline{\rho}_t)\big)\nabla_{\overline{\boldsymbol{\theta}}}\{\overline{a}^t\sigma_{\overline{s}}(\langle \overline{\boldsymbol{u}}^t, \boldsymbol{z}\rangle)\}\Big] \,.$$

$$\hat{f}_{\mathsf{NN}}(\boldsymbol{z};\overline{\rho}_t) = \int \overline{a}^t \mathbb{E}_G[\sigma(\langle \overline{\boldsymbol{u}}^t, \boldsymbol{z}\rangle + \overline{s}^t G)]\overline{\rho}_t(\overline{\boldsymbol{\theta}}^t) \,,$$

  from initialization $\overline{a}^0 \sim \mu_a$, $\overline{\boldsymbol{u}}^0 = \boldsymbol{0}$ and $\overline{s}^0 = \kappa$.

  - Gradient flow to learn $h_*(\boldsymbol{z})$ with effective 2-layer NN $\hat{f}_{\mathsf{NN}}(\boldsymbol{z};\overline{\rho}_t)$.

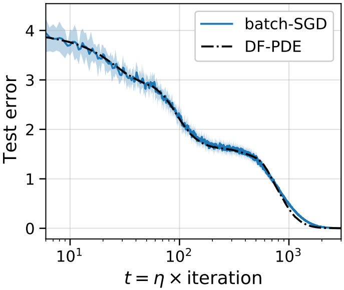
- As $d \to \infty$, MF dynamics concentrates on an effective dynamics over summary statistics of the weights and of the data.

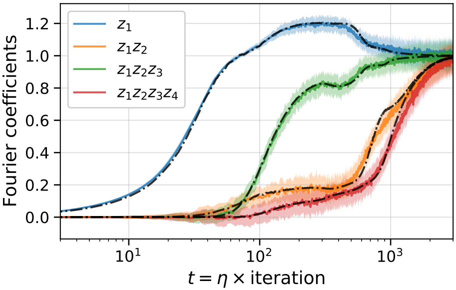
  $\Longrightarrow$ Wasserstein gradient flow on $\overline{\rho}_t \in \mathcal{P}(\mathbb{R}^{P+2})$ instead of $\mathcal{P}(\mathbb{R}^{d+1})$.

Numerical illustration

$d = 100$, $M = 100$:

$$h_*(z) = z_1 + z_1 z_2 + z_1 z_2 z_3 + z_1 z_2 z_3 z_4$$





Learning in $\Theta(d)$ iterations

- [Abbe, Boix-Adsera, M.,'22] With probability at least $1 - 1/M$:

$$\sup_{k \in \{0,\ldots,T\}} \left\| \hat{f}_{\mathsf{NN}}(\cdot; \boldsymbol{\Theta}^k) - \hat{f}_{\mathsf{NN}}(\cdot; \bar{\rho}_{k\eta}) \right\|_{L^2} \leq K e^{K(\eta T)^7} \Bigg[\underbrace{\sqrt{\frac{P}{d}}}_{d \to \infty} + \underbrace{\sqrt{\frac{\log(M)}{M}}}_{M \to \infty} + \underbrace{\sqrt{d\eta}}_{\eta \to 0} \Bigg]$$

- If DF-PDE achieves $O(\varepsilon)$-test error in $\overline{T}_* = \overline{T}(h_*, \varepsilon)$, so does SGD w.h.p. when

$$d \gtrsim C(\overline{T}_*) P/\varepsilon\,, \qquad M \gtrsim C(\overline{T}_*)/\varepsilon\,, \qquad \eta \lesssim d^{-1} \varepsilon / C(\overline{T}_*)\,,$$

Number of online SGD iterations (# samples) $T = C(\overline{T}_*) d/\varepsilon = \Theta(d)$.

- For which h_*, does the DF-PDE converge to zero?
(and therefore, h_* learned in $\Theta(d)$ steps in this regime)

Leap-1 functions

Fourier basis expansion of $h_* : \{\pm 1\}^P \to \mathbb{R}$ (with $\mathcal{Q}$ set of all $c_S \neq 0$, $S \subseteq \{1, \dots, P\}$)

$$h_*(z) = \sum_{S \in \mathcal{Q}} c_S \cdot \prod_{i \in S} z_i \,.$$

Leap-1 functions

$h_* : \{\pm 1\}^P \to \mathbb{R}$ is a *leap*-1 *function* if we can order its non-zero monomials $\mathcal{Q} = (S_1, \dots, S_r)$ such that for any $j \in [r]$, we have $|S_j \setminus (S_1 \cup \dots \cup S_{j-1})| \leq 1$.

E.g., leap-1 functions:

$$h_*(z) = z_1 + z_1 z_2 + z_1 z_2 z_3 + z_1 z_2 z_3 z_4,$$
$$h_*(z) = z_1 + z_1 z_2 + z_2 z_3 + z_3 z_4 + z_3 z_4 z_5.$$

E.g., "higher leap" functions:

$$h_*(z) = z_1 + z_1 z_2 z_3 + z_1 z_2 z_3 z_4,$$
$$h_*(z) = z_1 + z_1 z_2 + z_3 z_4 + z_3 z_4 z_5.$$

Leap-1 functions are learnable in $\Theta(d)$ steps

Theorem [Abbe, Boix-Adsera, Misiakiewicz,'22]

It is necessary and nearly sufficient* for h_* to be a leap-1 function in order for DF-PDE to converge to 0 test error**.

*Excludes a set of leap-1 functions $\left\{h_* = \sum_{S \in \mathcal{Q}} c_S \chi_S\right\}$ with $\{c_S\}_{S \in \mathcal{Q}}$ of Lebesgue-measure-0.
(This is unavoidable: DF-PDE does not converge for some degenerate leap-1 functions)

**For positive result: layerwise training. Train $\overline{\boldsymbol{u}}^t$ for $\overline{T}_1$ time, then $\overline{a}^t$ for $\overline{T}_2$ time.

- Leap-1 functions are essentially the functions that are learned in $\Theta(d)$ steps.

$$\underbrace{h_{*,1}(\boldsymbol{z}) = z_1 + z_1 z_2 + z_1 z_2 z_3}_{T = \Theta(d) \text{ SGD steps to learn}}, \qquad \underbrace{h_{*,2}(\boldsymbol{z}) = z_1 z_2 z_3}_{\text{needs } T \gg d \text{ steps}}.$$

Intuition

- Learning $h_*(z) = z_1 z_2$ with DF-PDE (recall $\overline{u}_1^0 = \overline{u}_2^0 = 0$):

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_1^t \approx \mathbb{E}_z[h_*(z)\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)z_1] = \mathbb{E}_z[z_2\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)] \propto \overline{u}_2^t\,,$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_2^t \approx \mathbb{E}_z[h_*(z)\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)z_2] = \mathbb{E}_z[z_1\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)] \propto \overline{u}_1^t\,.$$

  Hence dynamics is stuck at initialization $\overline{u}_1^t = \overline{u}_2^t = 0$.

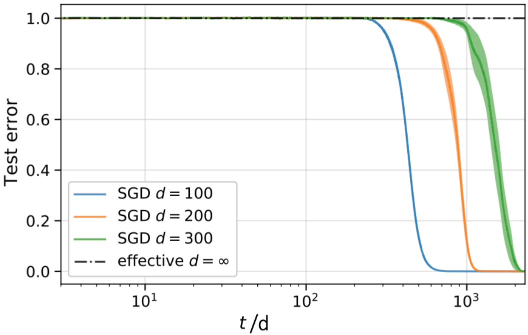
- Learning $h_*(z) = z_1 + z_1 z_2$ with DF-PDE:

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_1^t \approx \mathbb{E}_z[h_*(z)\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)z_1] = \mathbb{E}_z[(1+z_2)\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)] \propto 1 + \overline{u}_2^t\,,$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{u}_2^t \approx \mathbb{E}_z[h_*(z)\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)z_2] = \mathbb{E}_z[(z_1 z_2 + z_1)\sigma'(\langle \overline{\boldsymbol{u}}^t, z\rangle)] \propto \overline{u}_1^t\overline{u}_2^t + \overline{u}_1^t\,.$$

  Hence low degree term allows the dynamics to escape saddle.

Higher leap functions:

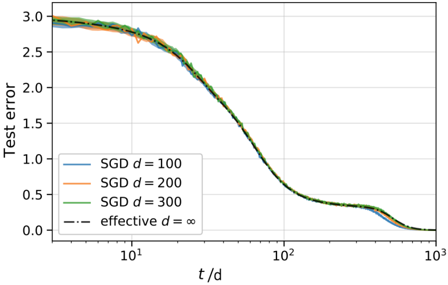
$$h_{*,2}(z) = z_1 z_2 z_3 \, .$$

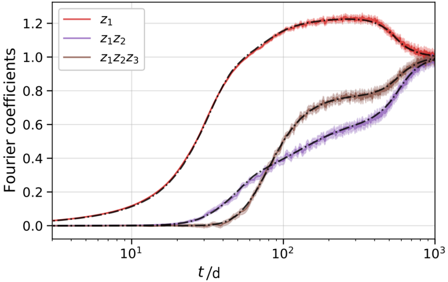


Effective dynamics initialized at a saddle point (SGD needs $T \gg d$ to escape).

Leap-1 functions:

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



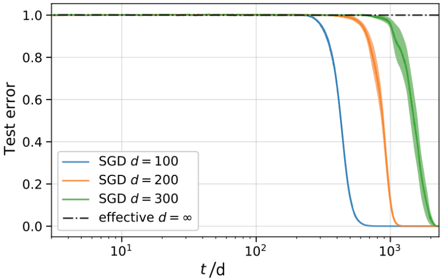


Low-degree terms allow escaping the saddle point.

2 What about higher leap functions?

Escaping the saddle

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



Theorem [Abbe, Boix-Adsera, Misiakiewicz,'23]

For $h_*(z) = z_1 \dots z_k$ ("leap-k" function), SGD need $\widetilde{O}(d^{k-1})$ steps to escape the saddle and fit the function.

Saddle: SGD slowly aligns w's with the k coordinates. k captures saddle complexity
$k =$ "information exponent" [Ben Arous, Gheissari, Jagannath,'21]

"Leap complexity"

$$h_*(z) = \sum_{S \in \mathcal{Q}} c_S \cdot \prod_{i \in S} z_i \,.$$

Leap complexity

We define the leap complexity of h_* as

$$\mathrm{Leap}(h_*) := \min_{\pi \in \Pi_{|\mathcal{Q}|}} \max_{i \in |\mathcal{Q}|} \left| S_{\pi(i)} \setminus \left(S_{\pi(1)} \cup \ldots \cup S_{\pi(i-1)} \right) \right| .$$

In words, $\mathrm{Leap}(h_*) \leq k$ iff we can order its non-zero monomials in a sequence such that each time a monomial is added, the support of h_* grows by at most k new coordinates.

$$\mathrm{Leap}(z_1 + z_1 z_2 + z_1 z_2 z_3 + z_1 z_2 z_3 z_4) = 1 \,, \qquad \mathrm{Leap}(z_1 + z_2 + z_2 z_3 z_4) = 2 \,,$$

$$\mathrm{Leap}(z_1 + z_1 z_2 z_3 + z_2 z_3 z_4 z_5 z_6 z_7) = 4 \,, \qquad \mathrm{Leap}(z_1 z_2 z_3 + z_2 z_3 z_4) = 3 \,,$$

A general conjecture

Conjecture

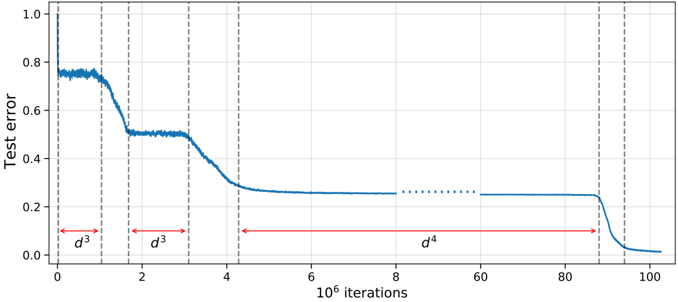
For all but a measure-0 set of target functions h_*, online SGD requires

$$\widetilde{\Theta}\left(d^{(\mathrm{Leap}(h_*)-1)\vee 1}\right) \text{ steps to learn.}$$

- Expect to hold for multilayer fully-connected NNs.
- Similar definition of Leap/conjecture for isotropic Gaussian data $\boldsymbol{x} \sim \mathsf{N}(0, \mathbf{I}_d)$. (more natural setting: can remove measure-0 set by considering an "isotropic" version of the leap)
- Total time complexity $\widetilde{\Theta}\left(d^{\mathrm{Leap}(h_*)\vee 2}\right)$ matches lower bound of a large class of algorithms: the correlation statistical query (CSQ) algorithms.

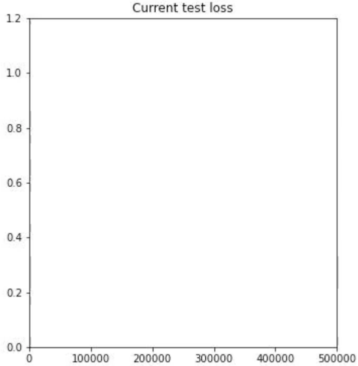
Saddle-to-saddle dynamics

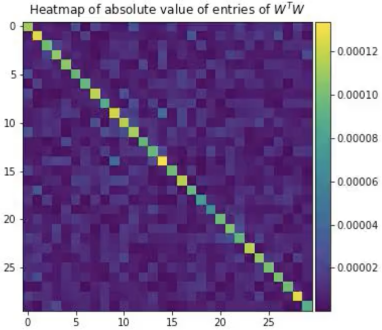
$$h_*(z) = z_1 + z_1 z_2 \cdots z_5 + z_1 z_2 \cdots z_9 + z_1 z_2 \cdots z_{14} \,.$$



Picture: SGD sequentially aligns the weights with the sparse support with a saddle-to-saddle dynamics.

$$h_*(z) = \frac{1}{\sqrt{3}}\left(z_1 + z_1 z_2 z_3 z_4 + z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8\right).$$





$d = 30$, covariance of first layer weights during training.

Partial proof of the conjecture

- Difficulties:
  - For $T \gg d$, cannot use PDE approximation ($e^{\eta T}$ propagation of error).
  - Requires to control a multiphase trajectory.

- Proof for $\boldsymbol{x} \sim \mathsf{N}(0, \mathbf{I}_d)$ and

$$h_*(\boldsymbol{z}) = z_1 z_2 \cdots z_{P_1} + z_1 z_2 \ldots z_{P_2} + \ldots + z_1 z_2 \cdots z_{P_L} ,$$

  with following modifications of SGD:
  - Layerwise training: first $\boldsymbol{w}_j$ for T_1 steps and then a_j for T_2 steps.
  - $\ell_\infty + \ell_2$ projection on $\boldsymbol{w}_j$.

- Show:
  - If $T_1 = d^{\mathrm{Leap}(h_*)-1} \log(d)^C$, can fit with $T_2 = \Theta(1)$.
  - If $T_1 \leq d^{\mathrm{Leap}(h_*)-1} / \log(d)^C$, cannot fit even with $T_2 = \infty$.
  - More precise theorem for saddle-to-saddle with increasing leaps.

General picture

When learning multi-index polynomials h_*:

- Kernel methods require $\Theta(d^{\mathrm{Degree}(h_*)})$ samples.

- Online SGD on NNs: $n = \widetilde{\Theta}\left(d^{(\mathrm{Leap}(h_*)-1)\vee 1}\right)$ samples/steps.

  $$\text{Typically:} \quad \mathrm{Leap}(h_*) \ll \mathrm{Degree}(h_*)$$

  (In fact, $\mathrm{Leap}(h_*) = 1$ a.s. on Fourier coeffs.)

- SGD picks up the support sequentially with a saddle-to-saddle dynamics.

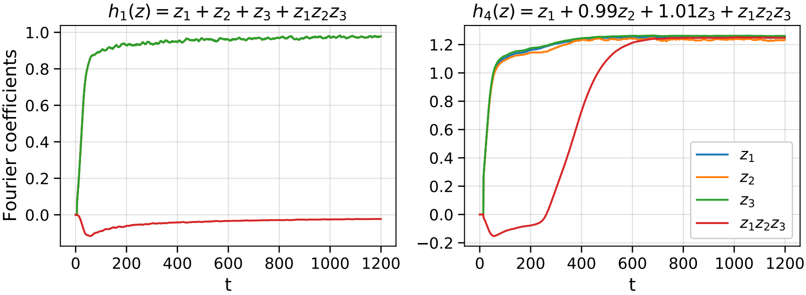
- Implement "adaptive"/"curriculum" learning: first learn low-degree monomials, which in turn, makes learning higher-degree monomials easier.

$h_*(z) =$	$z_1 \cdots z_{2k}$	$z_1 \cdots z_k + z_1 \cdots z_{2k}$	$z_1 + z_1 z_2 + \ldots + z_1 \cdots z_{2k}$
Kernels	$\Omega(d^{2k})$	$\Omega(d^{2k})$	$\Omega(d^{2k})$
SGD on NN	$\tilde{\Theta}(d^{2k-1})$	$\tilde{\Theta}(d^{k-1})$	$\Theta(d)$

Thank you!

Degenerate Leap-1 function

$d = 100$, $M = 100$:



$h_*(z) = z_1 + z_2 + z_3 + z_1z_2z_3$: we have $u_1^t = u_2^t = u_3^t$ during the dynamics.

The Gaussian case

$$h_*(z) = \sum_{S \in \mathbb{Z}^P} \hat{h}_*(S)\chi_S(z), \qquad \chi_S(z) = \mathrm{He}_{S_i}(z_i)\,.$$

For h_* with on-zero basis elements given by the subset $\mathcal{S}(h_*) := \{S_1, \ldots, S_m\}$

$$\mathrm{Leap}(h_*) := \min_{\pi \in \Pi_m} \max_{i \in [m]} \left\| S_{\pi(i)} \setminus \cup_{j=0}^{i-1} S_{\pi(j)} \right\|_1 ,$$

where

$$||S_{\pi(i)} \setminus \cup_{j=0}^{i-1} S_{\pi(j)}||_1 := \sum_{k \in [P]} S_{\pi(i)}(k) \mathbb{1}\{S_{\pi(j)}(k) = 0, \forall j \in [i-1]\}$$

Examples:

$$\mathrm{Leap}(\mathrm{He}_k(z_1)) = \mathrm{Leap}(\mathrm{He}_1(z_1)\mathrm{He}_1(z_2)\cdots\mathrm{He}_1(z_k)) = k\,,$$
$$\mathrm{Leap}(\mathrm{He}_{k_1}(z_1) + \mathrm{He}_{k_1}(z_1)\mathrm{He}_{k_2}(z_2) + \mathrm{He}_{k_1}(z_1)\mathrm{He}_{k_2}(z_2)\mathrm{He}_{k_3}(z_3)) = \max(k_1, k_2, k_3)\,,$$
$$\mathrm{Leap}(\mathrm{He}_2(z_1) + \mathrm{He}_2(z_2) + \mathrm{He}_2(z_3) + \mathrm{He}_3(z_1)\mathrm{He}_8(z_3)) = 2\,.$$

IsoLeap

Def of Leap depends on the specific coordinate basis used in the expansion.

Rotational symmetry of Gaussian distribution: use "isotropic leap":

$$\mathrm{isoLeap}(h_*) = \max_{R \in \mathcal{O}_P} \mathrm{Leap}(h_*, R)\,,$$

E.g., $h_*(z) = z_1 + z_2 + z_1 z_2$: leap-1 in this basis.

Take instead $(u_1, u_2) \to (z_1 + z_2, z_1 - z_2)/\sqrt{2}$

$$h_*(z) = u_1 + \mathrm{He}_2(u_1)/\sqrt{8} - \mathrm{He}_2(u_2)/\sqrt{8}\,.$$

Hence $\mathrm{isoLeap}(h_*) = 2$.

Other example

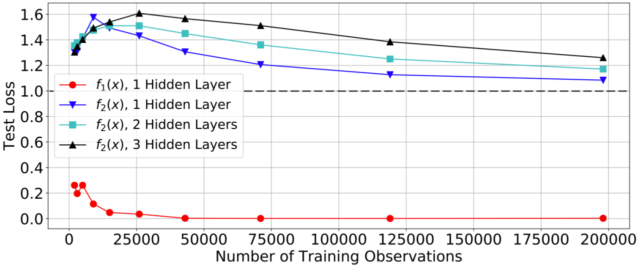
Problem: learning ridge functions with deep neural networks (DNNs).

$$(\boldsymbol{x}_i, y_i) \text{ iid with } y_i = f_s(\langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle) \text{ and } \boldsymbol{x}_i \sim \text{Unif}([\pm\sqrt{3}]^d), \ \ \|\boldsymbol{\theta}\|_2 = 1,$$

$$f_1(x) = \frac{\mathtt{tanh}(x)}{0.628}, \qquad f_2(x) = \frac{1}{0.1275}\Big(\mathtt{tanh}(x) - 3.422\ \mathtt{tanh}^3(x) + 2.551\ \mathtt{tanh}(x)^5\Big).$$

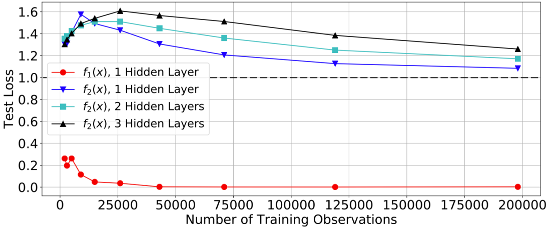
[Schmidt-Hieber,'17] DNNs can estimate both at nearly parametric rate $\log^2 n / n$.

Take $d = 500$ and train DNNs with SGD (100 neurons per hidden layer):



[AoS discussion, Ghorbani, Mei, **Misiakiewicz**, Montanari, 2020].

$$f_1(x) = \frac{\tanh(x)}{0.628}, \qquad f_2(x) = \frac{1}{0.1275}\left(\tanh(x) - 3.422\,\tanh^3(x) + 2.551\,\tanh(x)^5\right).$$



[Abbe, Boix-Adsera, Misiakiewicz,'23]

- $f_1(\langle \boldsymbol{\theta}, \cdot \rangle)$ leap-1 function: $\Theta(d)$ steps.
- $f_2(\langle \boldsymbol{\theta}, \cdot \rangle)$ leap-5 function: $\widetilde{\Theta}(d^4)$ steps.
- Take instead $f_3(\langle \boldsymbol{\theta}, \cdot \rangle)$ leap-3 fct

$$f_3(x) = \frac{1}{0.2292}\left(\tanh(x) - 1.4289\,\tanh^3(x)\right).$$