Lecture 9: Learning Multi-index functions

In this lecture, we consider learning a specific class of functions: multi-index fcts, that is fcts that only depend on the input data through a low-dimensional projection.

The goal is to get more insights on

- When and how can NNs overcome the curse of dimensionality?
- How do they adapt dynamically to low-dimensional support?
- How do they compare to other methods? Optimal algorithms?

Data distribution: "multi-index functions"

$$x \sim \mathrm{Unif}\left(\mathbb{S}^{d-1}(\sqrt{d})\right) \qquad y = f_*(x) + \varepsilon$$

(ε: mean 0, indep noise, e.g. $N(0,\sigma^2)$)

$$f_*(x) = g(U_*^T x) \qquad g: \mathbb{R}^\Delta \to \mathbb{R}$$

$$U_* \in \mathbb{R}^{d\times\Delta} \qquad U_*^T U_* = I_\Delta$$

$$= g(\langle u_1, x\rangle, \dots, \langle u_\Delta, x\rangle)$$

Δ "indices"

(The support U_* is unknown)

We are interested in the high-dimensional regime where g is fixed (in particular s) while $d \to \infty$.

Learn this data with a 2-layer NN.

We saw "three paradigms" of learning:

(1) Classical ERM with explicit regularization
→ assume algo output approximate minimizer of ERM

(2) Kernel methods
→ NNs trained in lazy regime

(3) Feature learning
→ NNs trained non-linearly (e.g. MF regime)
↳ adapt "representation" to data

In these three cases: sample complexity ?
computational complexity ? } → as $d \to \infty$

1 ERM with infinite-width 2 layer NN

"Breaking the curse of dimensionality with convex NNs"
– Francis Bach, 2017.

Model: $f(x) = \frac{1}{M} \sum_{j=1}^{M} a_j \sigma(\langle w_j, x \rangle + b_j)$

$\sigma(t) = \max(t, 0)$ RELU

Take $z = (x, \sqrt{d}) \in \mathbb{R}^{d+1}$ $v = (w, \frac{b}{\sqrt{d}}) \in \mathbb{R}^{d+1}$

$\sigma(\langle w_j, x \rangle + b_j) = \sigma(\langle v_j, z \rangle)$

$\mu(dv) = \text{Unif}(S^d)$ $V := S^d$

Lecture 2: set of ∞-width 2-layer NN

$$F_1(R) = \left\{ f(x) = \int_V \sigma(v^T z)\, a(v)\, \mu(dv) : \int_V |a(v)|\, \mu(dv) \le R \right\}$$

In fact, closure of $\mathcal{F}_1(R)$ can be written as

$$\overline{\mathcal{F}_1}(R) = \left\{ f(x) = \int_V \sigma(v^T x)\, \nu(dv) : \|\nu\|_{TV} \leq R \right\}$$

where $\|\nu\|_{TV} = \nu_+(V) + \nu_-(V)$

↳ signed measure

We saw that the Rademacher complexity

$$\boxed{\mathrm{Rd}_n\left(\overline{\mathcal{F}_1}(R)\right) \leq C R \sqrt{\frac{d}{n}}}$$

We can define a functional norm:

$$\|f\|_{\mathcal{F}_1} = \inf\left\{ \|\nu\|_{TV} : f(x) = \int_V \sigma(v^T x)\, \nu(dv) \right\}$$

ERM

$$\hat{f} = \underset{\|f\|_{\mathcal{F}_1} \leq R}{\operatorname{argmin}} \quad \hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2$$

Test error :

$$R(\hat{f}) = \mathbb{E}\left[(\hat{f}(x) - f_*(x))^2\right]$$

$$\leq \underbrace{\inf_{\|f\|_{\mathcal{F}_1} \leq R} \|f - f_*\|_{L^2}^2}_{\text{Approx error}} + \underbrace{2 \sup_{\|f\|_{\mathcal{F}_1} \leq R} |\hat{R}_n(f) - R(f)|}_{\text{generalization error}}$$

Approximation error :

Prop [Bach, 2017] For $D \geq 1$ and $R \geq C_d$ and any fct $\varphi : \mathbb{R}^D \to \mathbb{R}$ s.t. for all $\|x\|_2, \|y\|_2 \leq B$

$$|\varphi(x)| \leq L \qquad |\varphi(x) - \varphi(y)| \leq \frac{L}{B}\|x - y\|_2$$

There exists f with $\|f\|_{\mathcal{F}_1} \leq R$ and

$$\sup_{\|x\|_2 \leq B} |\varphi(x) - f(x)| \leq C_d L \left(\frac{L}{R}\right)^{\frac{2}{D+1}} \log\left(\frac{R}{L}\right)$$

In our case : $D = s$ & we can use $L = O(1)$ (using tail bounds)

For generalization error: we saw how to bound it from $Rd_n(F)$ using Talagrand contraction inequality for Lipschitz loss

Here: squared loss.

→ assume $\|y\|_\infty \leq C$

$$|f(x)| = \left| \int (\langle v, x \rangle)_+ \nu(dv) \right| \leq C\sqrt{d}\, \|\nu\|_{TV}$$

Hence $|\ell(y,f) - \ell(y,f')| \leq C(1 + R\sqrt{d})\,|f - f'|$

$$\mathbb{E}\left[\sup_{\|f\|_{\mathcal{F}_1} \leq R} \left| \hat{R}_n(f) - R(f) \right| \right] \leq \frac{C}{\sqrt{n}} + C(1+R\sqrt{d})\, Rd_n(\mathcal{F}_1(R))$$

$$\leq \frac{C}{\sqrt{n}} + C(1+R\sqrt{d})\, R\sqrt{\frac{d}{n}}$$

Putting these two together, we obtain:

$$\mathbb{E}[R(\hat{f})] \lesssim R^{-\frac{2}{s+1}} \log R + R^2 \frac{d}{\sqrt{n}}$$

→ take $R_* \simeq \left(\frac{\sqrt{n}}{d}\right)^{\frac{s+1}{2s+4}}$

we deduce

$$\mathbb{E}[R(\hat{\beta})] \lesssim \frac{\sqrt{d}}{n^{\frac{1}{2\delta+4}}} \log n$$

As long as $n \gtrsim d^{\delta+2}$ test error is vanishing

[Note that this is not an optimized proof]

↳ e.g with a little bit more care we can get

bound $\lesssim \left(\frac{d^{1+\frac{2}{\delta+1}}}{n}\right)^{\frac{1}{\delta+4}} \log^C n$ (can be probably improved)

This learning guarantee is for any $O(1)$-Lipschitz fct $f_*(x) = g(U_*^\top x)$ $U_* \in \mathbb{R}^{d\times D}$ as long as we have $\hat{\beta}$ a good enough approximate minimizer

Is there an efficient way of constructing approximate minimizers over F_1 ?

1st natural approach "random feature approximation"

$$\int \sigma(\langle v, z\rangle)\, a(v)\, \mu(dv) \to \frac{1}{M}\sum_{j=1}^{M} a_j\, \sigma(\langle v_j, z\rangle) =: f(z;a)$$

$$v_j \overset{iid}{\sim} \mu(dv)$$

$$\hat{f} = \operatorname*{argmin}_{a\in\mathbb{R}^M} \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(x_i;a)\right)^2 + \frac{\lambda}{M}\sum_{j=1}^{M}|a_j|$$

Approximation lower bound $d^{k+\delta} \le M \le d^{k+1-\delta}$

$$R(\hat{f}) \ge \inf_{a\in\mathbb{R}^M} \mathbb{E}\left[\left(f_*(x) - f(x;a)\right)^2\right] = \|P_{>k} f_*\|_{L^2}^2$$

can only fit $P_{\le k}\, g(U_*^T x)$

best degree-k polynomial approx

When fitting $f_*(x) = (\langle w_*, x\rangle + b_*)_+$ $\quad \|w_*\|_2 = d^2$, $|b_*| \le 6d^4+1$

[Yehudai, Shamir, '19]

Then test error must be $\ge \frac{1}{50}$ unless $M = e^{\Omega(d)}$ or $\|a\|_1 = e^{\Omega(d)}$

This is because we are trying to fit $\varphi(\langle w_*, x\rangle)$ with $\sigma(\langle w_j, x\rangle)$ and in high-dim $|\langle w_j, w_k\rangle| \lesssim \frac{1}{\sqrt{d}}$ for all $j \in [M]$ w.h.p unless M superpolynomial in d.

2nd natural approach adding one neuron at a time

Rmk: [We can choose $\hat{f}$ supported on $(n+1)$ neurons] (a type of "representer" theorem)

$$\inf_{\|f\|_{F_1} \le R} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$= \inf \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 : \hat{y} \in \mathbb{R}^n, \ \gamma_1(\hat{y}) \le R \right\}$$

where

$$\gamma_1(\hat{y}) = \inf\left\{ \|\nu\|_{TV} : \int \sigma(\langle v, z_i\rangle) \nu(dv) = \hat{y}_i \right\}_{i=1,\dots,n}$$

By Carathéodory theorem*, there optimal function can be decomposed into at most $(n+1)$ $\sigma(\langle v_j, z\rangle)$ i.e. we can write

[* if $x \in \mathrm{Conv}(P) \subset \mathbb{R}^n$ then x can be written as combinat° of $n+1$ elements of P]

$$\hat{f}(x) = \sum_{j=1}^{m+1} a_j \sigma(\langle v_j, x \rangle) \qquad \text{with} \sum_{j=1}^{m+1} |a_j| \leq R$$

Of course, we do not know these $\{v_j\}_{j=1}^{m+1}$ in advance

Instead of choosing v_j's at random as above, maybe we can choose them one at a time, that makes biggest progress in decreasing the train error.

Algo to minimize a functional $\Psi(f)$ over $f \in L^2(\rho)$ where ρ is some probe measure on $\mathcal{X}$

Conditional gradient algo (Frank-Wolfe algorithm):

Assume Ψ is convex and L-smooth and denote $\delta\Psi$ its functional gradient

$$0 \leq \Psi(f) - \Psi(h) - \langle f - h, \delta\Psi \rangle_{L^2(\rho)} \leq \frac{L}{2} \|f - h\|^2_{L^2(\rho)}$$

FW algo: iterative algo

- initialization $f_0 \in \overline{F}_1(R)$
- $\overline{f}_t \in \underset{f \in \overline{F}_1(R)}{\operatorname{argmin}} \langle f, \delta\Psi(f_t) \rangle_{L^2(e)}$

$$f_{t+1} := (1 - c_t) f_t + c_t \overline{f}_t$$

Note that $f_t \in \overline{F}_1(R)$ for all t

If we choose $c_t = \frac{2}{t+1}$, it is known that we have CV rate:

$$\Psi(f_t) - \inf_{f \in \overline{F}_1(R)} \Psi(f) \leq \frac{2L}{t+1} \sup_{f, h \in \overline{F}_1(R)} \|f - h\|^2_{L^2(e)}$$

Using that $\|f\|_\infty \leq C\sqrt{d}\, \|f\|_{F_1}$, we get

$$\boxed{\Psi(f_t) - \inf_{f \in \overline{F}_1(R)} \Psi(f) \leq \frac{C d R^2 L}{t+1}}$$

So if we can solve the update problem, this will CV to optimum

$$\langle f, g \rangle_{L^2(\rho)} = \int_V \left(\int_{\mathcal{X}} \sigma(\langle z, v \rangle)\, g(x)\, \rho(dx) \right) \nu(dv)$$

$$\leq \|\nu\|_{TV} \cdot \max_{v \in V} \left| \int_{\mathcal{X}} \sigma(\langle z, v \rangle)\, g(x)\, d\rho(x) \right|$$

Hence $$\max_{\|f\|_{F_1} \leq R} \langle f, g \rangle_{L^2(\rho)} = R \max_{v \in V} \left| \int_{\mathcal{X}} \sigma(\langle z, v \rangle)\, g(x)\, d\rho(x) \right|$$

which is achieved by taking $\pm R \cdot \sigma(\langle \cdot, v \rangle)$ where v is a maximizer of RHS.

In our case, $\mathcal{X} = \{x_1, \ldots, x_n\}$ and ρ is uniform measure

$$\Psi(f) = \hat{R}_n(f)$$

$$\nabla \Psi(f) = \frac{2}{n} \begin{pmatrix} f(x_1) - y_1 \\ \vdots \\ f(x_n) - y_n \end{pmatrix}$$

Therefore in our case, it reduces to solving the problem

$$\operatorname*{argmax}_{\|v\|_2 \leq 1} \sum_{i=1}^{n} a_i (v^T z_i)_+$$

This is a NP-hard problem even to solve approximately

Rmk: This is shown by doing a reduction to the problem of Maximin Agreement for Halfspaces problem

For $(x_1^{(1)}, \dots, x_{n_+}^{(1)})$ $(x_1^{(2)}, \dots, x_{n_-}^{(2)}) \in \{\pm 1\}^d$

$$M(w,a) = \sum_{i=1}^{n_+} \mathbb{1}[\langle w, x_i^{(1)} \rangle - a > 0] + \sum_{i=1}^{n_-} \mathbb{1}[\langle w, x_i^{(2)} \rangle - a < 0]$$

Then it is NP-hard to distinguish between the two cases:

(1) $\exists\, w, a$ s.t. $M(w,a) \geq (1-\varepsilon)(n_+ + n_-)$

(2) For any w, a: $M(w,a) \leq (\frac{1}{2} + \varepsilon)(n_+ + n_-)$

↑ classified at random

Rmk: Does that mean FW algo will not work?

(1) We do not need to solve exactly the update problem possible there is a convex relaxation [Open problem]

(2) NP-hard → but might still be tractable on random data [Open]

Two natural approaches failed. Is F_1-minimization intrinsically hard? (i.e, we can't hope to have a polynomial time algo that is guaranteed to minimize ERM problem with F_1)

See next lecture!

* Optimization hardness: getting $O(n^{-C})$ approximate minimizer is NP-hard
  [Celentano, M., Montanari, 2021]
* Learning hardness: if we could solve F_1-ERM then we could learn intersection of half spaces (under strong random CSP assumption, this is impossible)
  [Neyshabur, Tomioka, Srebro, 2015]

(More next lecture!)

Summary: F_1-ERM breaks the curse of dim when learning any Lipschitz multi-index fct in sample complexity. However, no polynomial time algo to construct $\hat{f}$

2 Kernel methods

In the lazy regime, 2-layer NNs behave as kernel method with inner-product kernel.
(For simplicity, we take all biases $b_j = b$)

We saw in lecture 6 that if $d^{k+\delta} \le n \le d^{k+1-\delta}$

$$R(\hat{f}) = \| P_{>k} f_* \|_{L^2}^2$$ ← only fit d° k polynomial approx

More generally, using the dimension lower bound, when learning multi-index model, with data uniform on the sphere, no kernel method can do better

Rmk: [Learning with F_1 vs F_2 regularization]

$$F_1(R) = \left\{ f(x,a) = \int \sigma(\langle \omega, x \rangle)\, a(\omega)\, \mu(d\omega) \;:\; \int |a(\omega)|\, \mu(d\omega) \le R \right\}$$ (uniform on the sphere)

$$F_2(R) = \left\{ f(x,a) = \int \sigma(\langle \omega, x \rangle)\, a(\omega)\, \mu(d\omega) \;:\; \left| \int |a(\omega)|^2\, \mu(d\omega) \right|^{1/2} \le R \right\}$$

Similarly to F_1, we can define a functional norm

$$\|f\|_{F_2} = \inf \left\{ \|a\|_{L^2(\mu)} : f = \int \sigma(\langle \omega, x \rangle)\, a(\omega)\, \mu(d\omega) \right\}$$

Then $$\min_f \ \hat{R}_n(f) + \lambda \|f\|_{F_2}$$

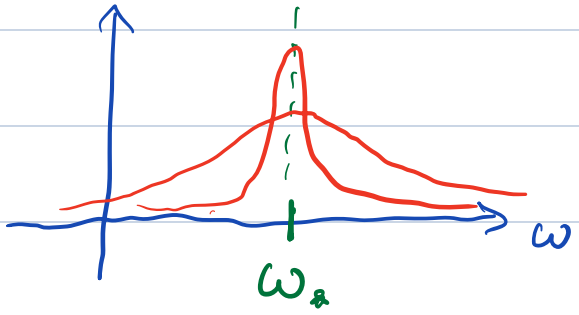
is a kernel method associated to the kernel

$$K(x_1, x_2) = \int \sigma(\langle \omega, x_1 \rangle)\, \sigma(\langle \omega, x_2 \rangle)\, \mu(d\omega)$$

and F_2 is an RKHS with RKHS-norm $\|f\|_{F_2}$

So: when $+\lambda \|f\|_{F_1}$ → learn
$+\lambda \|f\|_{F_2}$ → fail to learn
} multi-index fcts

Some heuristic intuition: if we want to learn $\sigma(\langle \omega_*, x \rangle)$
then to get a good approx using $a(\omega)\mu(d\omega) \to \delta_{\omega_*}$



sequence of (positive) $a_k(\omega)\mu(d\omega) \to \delta_{\omega_*}$

s.t. $\int |a_k(\omega)| \mu(d\omega) = 1$

but $\int |a_k(\omega)|^2 \mu(d\omega) \nearrow \infty$

We can approximate δ_{ω_*} with $\|a_k\|_{L^1} = 1$

but we must have $\|a_k\|_{L^2} \nearrow \infty$

worse & worse generalization error

(3) Feature learning

Consider: $f(x;\Theta) = \sum_{j=1}^{M} a_j \sigma(\langle w_j, x\rangle + b_j)$ ← Relu

and consider training $(a_j, w_j, b_j)_{j=1}^{M}$ using GD

on squared loss $\hat{R}(\Theta) = \frac{1}{n}\sum_{i=1}^{n} (y_i - f(x_i;\Theta))^2$

We follow [Domingo, Lee, Solkenolkokabi, 2022]

(paper is for Gaussian data, but can be modified for spherical data)

Assumption [Data] $x_i \sim N(0, Id)$ $\quad y_i = f_*(x_i) + \varepsilon_i$, $\varepsilon_i \sim \mathrm{Unif}(\pm 3)$

- $f_*(x) = g(\langle u_1, x\rangle, \dots, \langle u_s, x\rangle)$ → g: $d^\circ p$ polynomial; → orthogonal to $d^\circ 1$ polynomial in x (can be learned by preprocessing)
- $M = \mathbb{E}_x[\nabla^2 f_*]$ has rank s $\quad \kappa := \frac{1}{\sqrt{s}\,\sigma_{\min}(M)}$

Algo: 1) Initialization $\quad a_j^0 \sim \mathrm{Unif}(\pm 1)$, $a_j^0 = -a_{M-j}^0$ $\quad w_j^0 \sim N(0, \frac{Id}{d})$, $w_j^0 = w_{M-j}^0$

$b_j^0 = 0$

→ symmetric initialization s.t. $f(x, \Theta^0) = 0$

2) One gradient step on W (ℓ_2 regularization)

$$\begin{cases} W^{(1)} = W^{(0)} - \eta_1 \left[\nabla_W \hat{R}(\Theta^0) + \lambda_1 W^{(0)}\right] \\ a^{(1)} = a^{(0)} \end{cases}$$

3) Re-initialize $b_j \sim N(0,1)$

Train 2nd layer weights for $t = 2, \dots, T$

$$\begin{cases} a^{(t)} = a^{(t-1)} - \eta_2 \left[\nabla_a \hat{R}(\Theta^{(t)}) + \lambda_2 a^{(t-1)}\right] \\ W^{(t)} = W^{(t-1)} \end{cases}$$

Hyperparameters: $\eta_1 = \tilde{\Theta}(\sqrt{d}) \qquad \lambda_1 = \frac{1}{\eta_1}$

Thm: Assume $n = \tilde{\Omega}(d^2 \kappa^2 s)$ and $d = \tilde{\Omega}(\kappa s^{3/2})$

Then there exists η_2 and λ_2 sufficiently small,

$$T = \Theta\left(\frac{1}{\eta_2 \lambda_2}\right) \quad \text{s.t.} \quad \text{w.p} \geq 0.99$$

$$\mathbb{E}\left[\left|f(x; \Theta^{(T)}) - y\right|\right] - \xi \lesssim \sqrt{\frac{d s^p \kappa^{2p}}{n}} + \sqrt{\frac{s^p \kappa^{2p}}{M}} + \frac{1}{n^{1/4}}$$

Rmk: Learn with $n = \tilde{\Theta}_d(d^2)$ and $M = \Theta_d(1)$

Proof idea

$$\nabla_{w_j} \mathbb{E}\left[(y - f(x; \Theta^0))^2\right] = -2\,\mathbb{E}\left[f_*(x)\, \nabla_{w_j} f(x; \Theta^0)\right]$$

$$= -2 a_j\, \mathbb{E}\left[f_*(x)\, x\, \mathbb{1}_{w_j \cdot x \geq 0}\right]$$

$$\simeq -\frac{2}{\omega_j} a_j H w_j + \tilde{O}\left(\frac{\sqrt{\Delta}}{d}\right)$$

For $n = \tilde{\Omega}(d^2)$ population $\approx$ empirical gradient

$$W^{(1)} = W^{(0)} - \eta_1 \left(\nabla_W \hat{R}(W^{(0)}) + \frac{1}{\eta_1} W^{(0)} \right)$$

$$= -\eta_1 \nabla_W \hat{R}(W^{(0)})$$

$$\approx -\eta_1 M W^{(0)} \operatorname{diag}(a_j)$$

where $w_j^{(1)}$ are in the span of $U \in \mathbb{R}^{d \times s}$

$M W^{(0)}$ has singular value $\approx \frac{1}{\sqrt{d}}$

and we take $\eta_1 = O(\sqrt{d})$

Hence after this first step, we fix $W^{(1)}$ and get

$$f(x; a) = \sum_j a_j \sigma(\langle w_j, x \rangle + b_j)$$

s-dimensional span

↳ then optimizing over a_j: kernel methods with dim-s input

↳ can use approx + generalization type analysis (but for SGD) / does not depend on d

With this specific choice of GD: $W^{(1)}$ aligns with the support U after one step. Learn a good "feature representation" with $\sigma(\langle w_j, x \rangle)$ $w_j \in \text{span}(U)$

→ once learned the representation, problem becomes low-dimensional and learning is easy, e.g., linear regression on 2nd layer weights

Shortcomings of this analysis:

- $\mathbb{E}[\nabla_g^2]$ full rank : is it necessary?
  (e.g. not verified by $He_3(\langle u_1, x \rangle)$)

- $n = \Omega(d^2)$ is it necessary?
  ↳ e.g. $g(\langle u_1, x \rangle) = \langle u_1, x \rangle$ only need $n = \Theta(d)$

Can we get a more precise picture of how GD/SGD learn the low-dimensional support?

The staircase mechanism

[Abbe, Boix-Adsera, M., '22, '23]

For simplicity, consider $x \sim \mathrm{Unif}(\{\pm 1\}^d)$

Rmk: [Functions on the hypercube]

The Fourier-Walsh basis is an orthogonal basis of $L^2(\{\pm 1\}^d)$

$$\chi_S(x) = \underbrace{\prod_{i \in S} x_i}_{\text{"parity fct on subset } S\text{"}} \qquad S \subseteq [d] \quad \text{with } \chi_\emptyset(x) = 1$$

Indeed, we have

$$\mathbb{E}[\chi_S(x)\chi_{S'}(x)] = \mathbb{E}[\chi_{S \Delta S'}(x)] = \begin{cases} 0 & \text{if } S \Delta S' \neq \emptyset \\ 1 & \text{if } S \Delta S' = \emptyset \end{cases}$$

So any function $f : \{\pm 1\}^d \to \mathbb{R}$ can be decomposed as

$$f(x) = \sum_{S \subseteq [d]} \hat{f}(S)\, \chi_S(x) \qquad \hat{f}(S) = \mathbb{E}_x[f(x)\chi_S(x)]$$

We consider learning fcts with sparse support:

$$y_i = f_*(x_i) + \varepsilon_i$$

where $f_*(x) = g(z_1, z_2, \dots, z_s)$

with $(z_1, \dots, z_s) = (x_{i_1}, x_{i_2} \dots x_{i_s})$

unknown subset of s coordinates

From above:

$$g(z) = \sum_{S \subseteq [s]} \hat{g}(S) \prod_{i \in S} z_i$$

Motivating examples:

$$\begin{cases} g_1(z) = z_1 + z_1 z_2 + z_1 z_2 z_3 \\ g_2(z) = z_1 z_2 z_3 \end{cases}$$

Q: Which one is easier to learn with GD-trained NNs?

Rmk:

- Both are simple fcts that only depend on 3 coordinates and can be approximated easily by a 2-layer NN
- $\text{rank}(\mathbb{E}[\nabla^2 g_1(z)]) = 2 < 3$
  $\text{rank}(\mathbb{E}[\nabla^2 g_2(z)]) = 0 < 3$
  } both don't satisfy Damian et al condition

(Let's pretend z_i are iid $N(0,1)$ → theory apply to this case too)

To build intuition, we consider population gradients:

$$R(\Theta) = \mathbb{E}_x\left[(y - f(x;\Theta))^2\right]$$

$$f(x;\Theta) = \sum_{j=1}^{M} a_j \sigma(\langle w_j, x\rangle)$$

From earlier computation, for NNs to learn the target function, the weights W need to align with the support of f_*.
Let's freeze a_j and consider a few gradient step on w_j:

$$w_j^{(t+1)} = w_j^{(t)} - \eta \nabla_{w_j} R(\Theta^{(t)})$$

$$= w_j^{(t)} + \eta a_j \mathbb{E}\left[f_*(x)\, \sigma'(\langle w_j, x\rangle)\, x\right]$$

$$- \eta a_j \underbrace{\mathbb{E}\left[f(x;\Theta^t)\, \sigma'(\langle w_j, x\rangle)\, x\right]}$$

NN self interaction term that do not contain signal, $f(x;\Theta^0) \approx 0$
$\rightarrow$ at the beginning of the dynamics ≈ 0
and we will neglect it

Let's set $\eta a_j = 1$ for simplicity

At initialization, consider $w_j^0 \sim \mathrm{Unif}\left(\xi \pm \frac{1}{\sqrt{d}}\xi^d\right)$

(could be Gaussian too)

Lemma: At initialization

$$\mathbb{E}\left[\sigma'(\langle w_j^0, x\rangle) \prod_{i\in S} x_i\right] \approx \underbrace{\mathbb{E}\left[\sigma^{(|S|+1)}(G)\right]}_{=: \mu_{|S|+1}(\sigma)} \prod_{i\in S} (w_j^0)_i$$

"Hermite coeffs"

Proof sketch:

$$\mathbb{E}_{x_1}\left[\sigma'(\langle w, x\rangle) x_1\right] = \frac{1}{2}\left[\sigma'(w_1 + \langle w_{-1}, x_{-1}\rangle) - \sigma'(-w_1 + \langle w_{-1}, x_{-1}\rangle)\right]$$

$|w_1| = \frac{1}{\sqrt{d}}$ is small

$$\approx w_1 \, \sigma''(\langle w_{-1}, x_{-1}\rangle)$$

$$\frac{1}{\sqrt{d}} \sum_{j=1}^{d} u_j \Rightarrow N(0,1) \qquad \square$$

Let's use this lemma to get a heuristic picture of what happens at the beginning of the dynamics when learning either g_1 or g_2

Assume that $z = (x_1, x_2, x_3)$

(1) Learning $g_1(z) = x_1 + x_1 x_2 + x_1 x_2 x_3$

1st GD step: $w^{(1)} = w^{(0)} + \mathbb{E}\left[g_1(z)\, \sigma'(\langle w^{(0)}, x \rangle) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{pmatrix} \right]$

$$w_1^{(1)} = w_1^{(0)} + \mathbb{E}\left[\sigma'(\langle w^{(0)}, x \rangle)\, g(z)\, x_1 \right]$$

$$= w_1^{(0)} + \mathbb{E}\left[\sigma'(\langle w^{(0)}, x \rangle)\, (1 + x_2 + x_2 x_3) \right]$$

$$= \underbrace{w_1^{(0)}}_{\simeq \frac{1}{\sqrt{d}}} + \mu_1 + \underbrace{\mu_2 w_2^{(0)}}_{\simeq \frac{1}{\sqrt{d}}} + \underbrace{\mu_3 w_2^{(0)} w_3^{(0)}}_{\simeq \frac{1}{d}}$$

$$\boxed{w_1^{(1)} \simeq 1}$$

similarly $w_2^{(1)} = w_2^{(0)} + \mathbb{E}\left[\sigma'(\langle w^{(0)}, x \rangle)\, (x_2 + x_1 + x_1 x_3) \right]$

$$\boxed{w_2^{(1)} \simeq \frac{1}{\sqrt{d}}}$$

and $\boxed{\omega_3^{(1)} \asymp \frac{1}{\sqrt{d}}}$ $\boxed{\omega_j^{(1)} \asymp \frac{1}{\sqrt{d}}}$ for $j \geq 4$

After 1 step, $\omega^{(1)}$ align with 1st coordinate

2nd GD step:

$$\omega_2^{(2)} = \omega_2^{(1)} + \mathbb{E}\left[\sigma'(\langle \omega^{(1)}, x\rangle)(x_2 + x_1 + x_1 x_3)\right]$$

$$= \underbrace{\omega_2^{(1)}}_{\asymp \frac{1}{\sqrt{d}}} + \underbrace{\mu_2 \omega_2^{(1)}}_{\asymp \frac{1}{\sqrt{d}}} + \underbrace{\mu_2 \omega_1^{(1)}}_{\asymp 1} + \underbrace{\mu_3 \omega_1^{(1)} \omega_3^{(1)}}_{\asymp \frac{1}{\sqrt{d}}}$$

$\boxed{\omega_2^{(2)} \asymp 1}$

and $\boxed{\omega_3^{(2)} \asymp \frac{1}{\sqrt{d}}}$ $\boxed{\omega_j^{(2)} \asymp \frac{1}{\sqrt{d}}}$ $j \geq 4$

3rd GD step:

$$\omega_3^{(3)} \asymp \omega_3^{(2)} + \mathbb{E}\left[\sigma'(\langle \omega^{(2)}, x\rangle) x_1 x_2\right]$$

$$\asymp \underbrace{\omega_3^{(2)}}_{\asymp \frac{1}{\sqrt{d}}} + \mu_3 \underbrace{\omega_1^{(2)} \omega_2^{(2)}}_{\asymp 1}$$

$\boxed{\omega_3^{(3)} \asymp 1}$

GD sequentially aligns to the support, by using intermediary monomials to learn one coordinate at a time

$n = \Theta(d)$ samples is enough so that we can replace population by empirical gradient and have similar computation

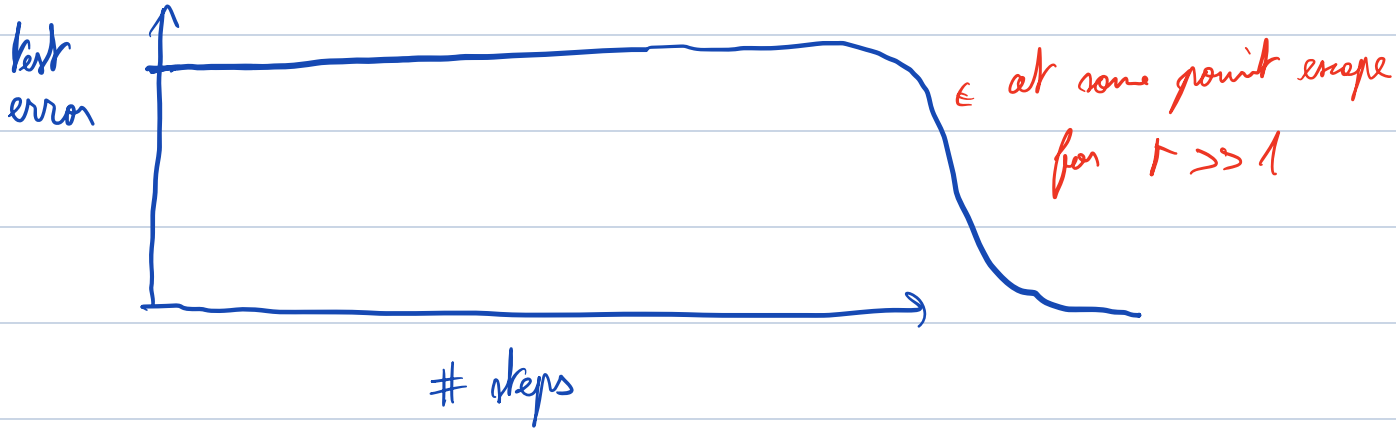
(2) Learning $g_2(x) = x_1 x_2 x_3$

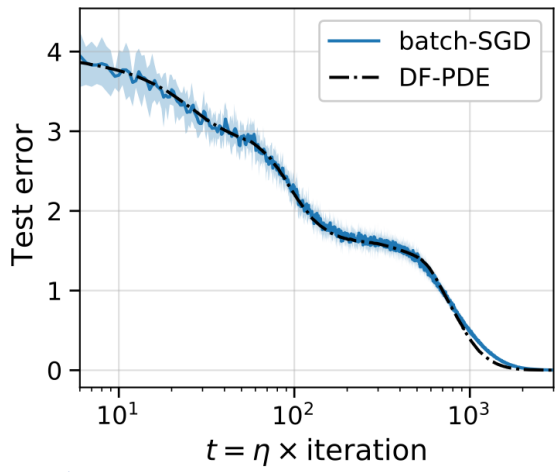
In that case, we don't have the intermediary "stairs" (monomials)

$$w_1^{(1)} = \underbrace{w_1^{(0)}}_{\simeq \frac{1}{\sqrt{d}}} + \underbrace{\mathbb{E}\left[\sigma'(\langle w^{(0)}, x\rangle)\, x_2 x_3\right]}_{\asymp \frac{1}{d}}$$

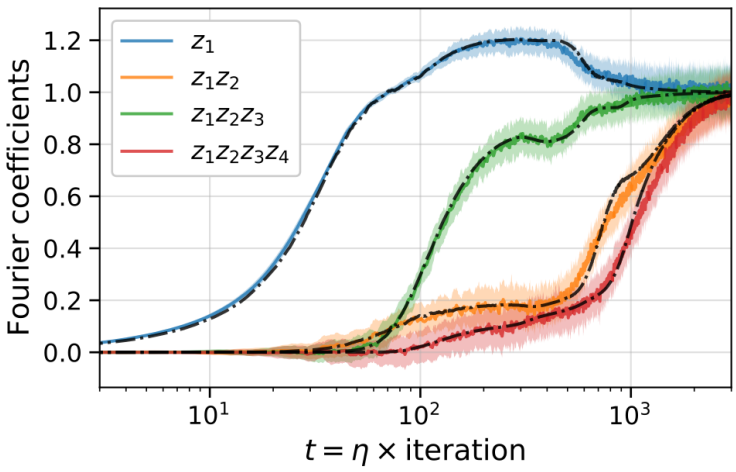
same for $w_2^{(1)}$ and $w_3^{(1)} \asymp \frac{1}{\sqrt{d}}$ and stay small for any constant # of steps

$\Rightarrow$ The dynamics is initialized near a saddle pt ($\nabla_w R(w^0) \approx 0$) and takes many more steps to escape





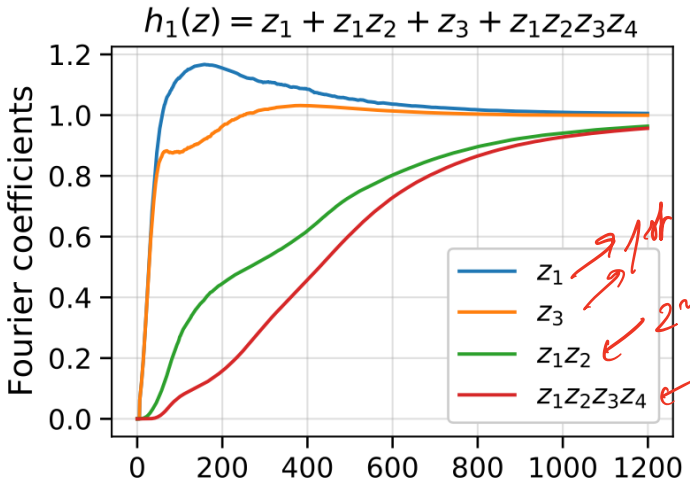
this is online SGD

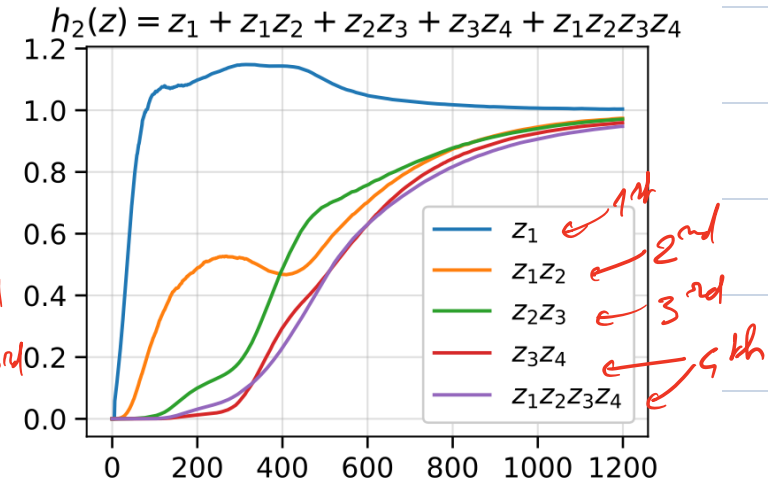


these are the Fourier coefficients of the NN throughout the dynamics, i.e.

$$\mathbb{E}\left[f_{NN}(x; \Theta^t) \prod_{i \in S} x_i \right]$$

Other examples:





We can show that $T = \tilde{O}_d(d)$ online SGD iteration are enough to learn these functions, hence $n = \tilde{O}_d(d)$ samples

Def: [Merged-Staircase fcts]

$$g(z) = \sum_{S \subset [s]} \hat{g}(S) \prod_{i \in S} z_i$$

Denote $\Sigma = \{S \subseteq [d] : \hat{g}(S)\}$. We say that Σ has MS property, if there exists an ordering

$$\Sigma = \{S_1, S_2, S_3, \dots, S_r\}$$

such that

$$|S_i \setminus \bigcup_{j=1}^{i-1} S_j| \leq 1 \qquad i \in [1, \dots, r]$$

What is we can order the non zero monomials such that we add at most one coordinate at a time

Following the heuristic argument from above, we expect these fct to be learnable in $\tilde{\Theta}(d)$ samples

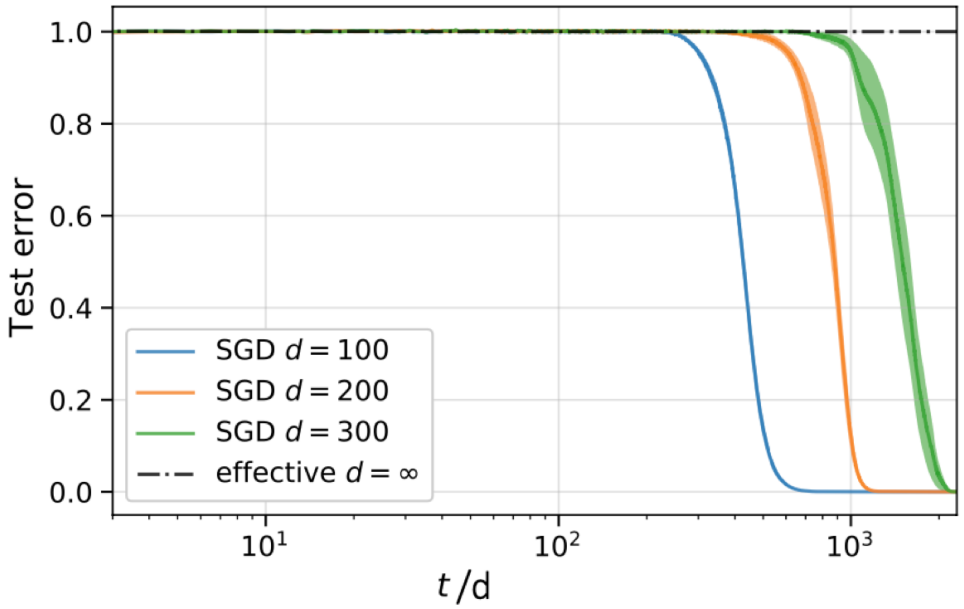
Thm: MSP fcts are learnable with $\Theta(d)$ online SGD steps, except for a Lebesgue measure-0 set of them

Conversely: if g is not MSP, then NNs trained in the mean-field regime will not learn g with $T = O(1)$ continuous time

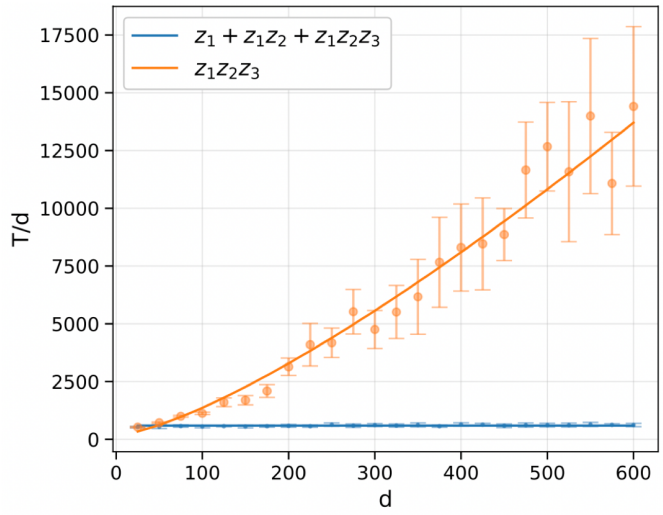
which corresponds to $\mathcal{O}(d)$ discrete SGD steps

What happens for non-MSP functions?

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

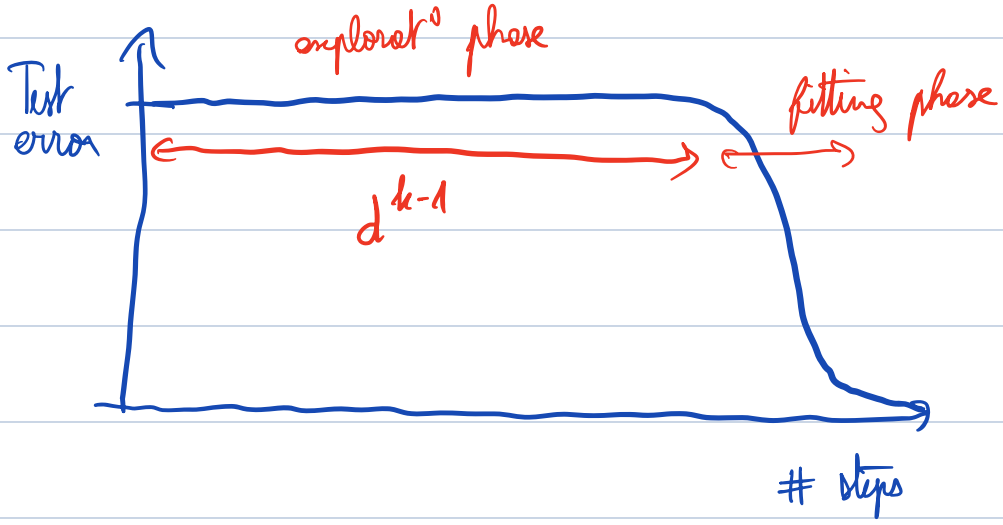


T = number of SGD steps to reach 0.05 test error.



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

E.g. $g_2(z) = z_1 z_2 z_3 \cdots z_k$

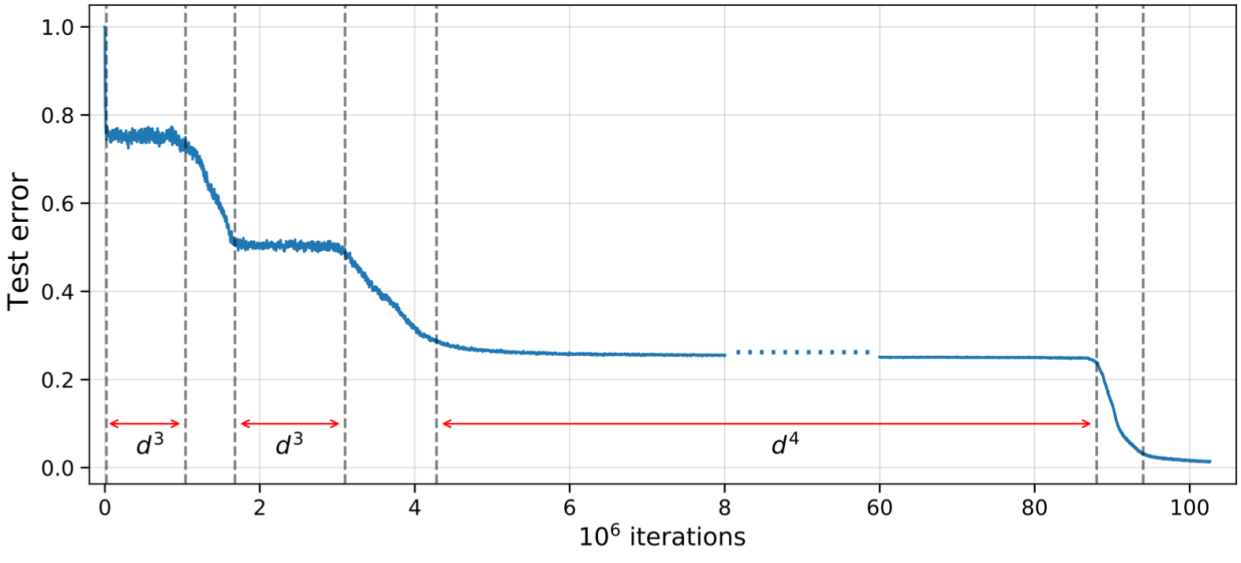


SGD requires $\tilde{O}(d^{k-1})$ step size to escape the saddle at initialization and align w's with the k coordinates in the support

→ as soon w's aligned with the support, NNs can quickly fit this fct using 2nd layer weights

More generally:

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



SGD sequentially aligns the weights with the sparse support with saddle-to-saddle dynamics, and requires $\widetilde{\Theta}(d^{k-1})$ SGD steps to align with $k \geq 2$ new coordinates at once

def: [Leap-complexity] g is a leap-k fct if it is the smallest integer such that we can order the non-zero monomials s.t.

$$|S_i / \bigcup_{j=1}^{i-1} S_j| \leq k$$

(MSP $\equiv$ leap-1 fcts)

Conjectural picture: online SGD with squared loss learns g with # steps

$$T = \begin{cases} \Theta(d) & \text{iff } g \text{ leap-1} \\ \Theta(d \log^c d) & \text{iff } g \text{ leap-2} \\ \Theta(d^{k-1} \log^c d) & \text{iff } g \text{ leap-}k \end{cases}$$

Summary:

* Functions of low-dimensional projection can be represented efficiently by 2-layer NNs

* If we could solve ERM efficiently, we could learn these fcts with poly(d) sample complexity
  (regardless of loop complexity of the fct)
  ↳ However, evidence that there cannot be an algo that provably solves F_1-ERM in poly(d) time

* Kernel methods cannot exploit this low-dimensional structure

* In contrast, NNs trained non-linearly: W align with the support, learn "good representation", and dynamics become low dimensional.
  ↳ learning is sequential, following a staircase
  ⟹ if function has high loop complexity, GD fail to learn it