Learning single-index models with neural networks

Denny Wu

dennywu@nyu.edu

Center for Data Science, New York University

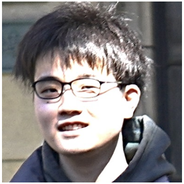
Center for Computational Mathematics, Flatiron Institute

Introduction

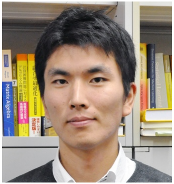
- ❒ [LOSW24] *Neural network learns low-dimensional polynomials near the information-theoretic limit.*
- ❒ [OSSW24] *Learning sum of diverse features: computational hardness and efficient gradient-based training for ridge combinations.*
- ❒ [OSSW24] *Pretrained transformer efficiently learns low-dimensional target functions in context.*

Jason D. Lee

Kazusato Oko



Yujin Song



Taiji Suzuki

Gaussian single-index model: $f_*(\boldsymbol{x}) = \sigma_*(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle)$, $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$.

- Requires learning the direction $\boldsymbol{\theta} \in \mathbb{R}^d$ and link function $\sigma_* : \mathbb{R} \to \mathbb{R}$.
  - Learning algorithm should adapt to **low-dimensional structure**.
- We assume σ_* is a polynomial with degree p and information exponent k.

Baseline I: information theoretic limit

Theorem ([Bach 17], [Barbier et al. 19], [Damian et al. 24]...)

Information theoretically, $n \asymp d$ samples are necessary and sufficient to learn f_*.

☹ For generic σ_*, algorithms may require **exponential compute** to achieve this.

Introduction: Single-index Model

Gaussian single-index model: $f_*(\boldsymbol{x}) = \sigma_*(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle), \ \boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$.

- Requires learning the direction $\boldsymbol{\theta} \in \mathbb{R}^d$ and link function $\sigma_* : \mathbb{R} \to \mathbb{R}$.
  - Learning algorithm should adapt to **low-dimensional structure**.
- We assume σ_* is a polynomial with degree p and information exponent k.

Baseline II: **complexity of non-adaptive (linear) estimators**

Theorem ([Ghorbani et al. 19], [Donhauser et al. 21], [Gavrilopoulos et al. 24]...)

Rotationally invariant kernel methods requires $n \gtrsim d^p$ *samples to learn* f_*.

Question: what is the statistical complexity of *adaptive* methods?

- **Example:** polynomial width neural network optimized by *gradient descent*.

Introduction: Information Exponent

Hermite expansion: $\sigma_*(z) = \sum_{i=0}^{\infty} \alpha_i^* \mathrm{He}_i(z)$, $\alpha_i^* = \mathbb{E}[\sigma_*(z)\mathrm{He}_i(z)]$.

Definition: information exponent [Ben Arous et al. 2021]

The information exponent of σ_* is defined as $k = \mathrm{IE}(\sigma_*) = \min\{k \in \mathbb{N}_+ : \alpha_k^* \neq 0\}$.

$$
\begin{aligned}
-\mathbb{E}[\nabla_{\boldsymbol{w}} \mathcal{L}(f_{\mathrm{NN}})] &\approx \mathbb{E}[\nabla_{\boldsymbol{w}}(f_{\mathrm{NN}}(\boldsymbol{x}) f_*(\boldsymbol{x}))] \\
&= \boldsymbol{\theta} \cdot \mathbb{E}[\sigma_*'(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle)\sigma'(\langle \boldsymbol{x}, \boldsymbol{w} \rangle)] + \boldsymbol{w} \cdot \mathbb{E}[\dots] \quad \textit{Stein's lemma} \\
&= \boldsymbol{\theta} \cdot \sum_{i=0}^{\infty} (i+1)^2 \alpha_{i+1}^* \beta_{i+1} \underbrace{\langle \boldsymbol{w}, \boldsymbol{\theta} \rangle^i}_{d^{-i/2} \text{ at initialization}} + \dots \quad \textit{Hermite expansion}
\end{aligned}
$$

- **Gradient concentration.** with high probability,

$$\left\| \mathbb{E}[\boldsymbol{x}\boldsymbol{\sigma}'(\langle \boldsymbol{x}, \boldsymbol{w} \rangle) f^*(\boldsymbol{x})] - \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{\sigma}'(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle) f^*(\boldsymbol{x}_i) \right\| \lesssim \sqrt{d/n}.$$

- $n = \Omega(d^k)$ samples required to achieve nontrivial concentration.

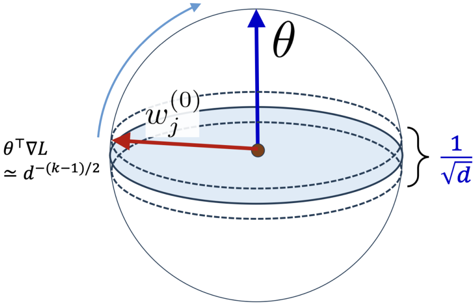
Introduction: Information Exponent

Hermite expansion: $\sigma_*(z) = \sum_{i=0}^{\infty} \alpha_i^* \mathrm{He}_i(z)$, $\alpha_i^* = \mathbb{E}[\sigma_*(z)\mathrm{He}_i(z)]$.

Definition: information exponent [Ben Arous et al. 2021]

The information exponent of σ_* is defined as $k = \mathrm{IE}(\sigma_*) = \min\{k \in \mathbb{N}_+ : \alpha_k^* \neq 0\}$.

Intuition: the amount of information in the gradient at *random initialization*.



- For $k > 1$, parameters are initialized at (approximate) saddle point .
- Most of the data is used to escape from the high entropy "equator" around initialization.

Introduction: Complexity of SGD Learning

Theorem ([Ben Arous et al. 21], [Bietti et al. 22], [Damian et al. 23]...)

A two-layer neural network optimized by (variants of) gradient descent can learn f_* with information exponent k using $n \gtrsim d^{\Theta(k)}$ samples.

- $k \leq p$ $\Rightarrow$ NN + gradient-based training outperforms *kernel model* ☺
- For large k, NN + GD cannot match the *information theoretic limit* ☹

Question: does information exponent capture the *computational hardness*?

Consider the gradient of expected squared loss for one neuron $f_{\boldsymbol{w}}(\boldsymbol{x})$:

$$\nabla_{\boldsymbol{w}} \mathbb{E}_{\boldsymbol{x},y}(f_{\boldsymbol{w}}(\boldsymbol{x}) - y)^2 \propto -\mathbb{E}_{\boldsymbol{x},y}[\underbrace{y \cdot \nabla_{\boldsymbol{w}} f_{\boldsymbol{w}}(\boldsymbol{x})}_{\text{correlational query}}] + \mathbb{E}_{\boldsymbol{x}}[\underbrace{f_{\boldsymbol{w}}(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{w}} f_{\boldsymbol{w}}(\boldsymbol{x})}_{\text{can be evaluated without } y}].$$

- **Idea:** count number of "accurate" correlational queries required by the algorithm.

Introduction: Statistical Query Lower Bounds

- **Statistical query (SQ).** Algorithm has access to "noisy" version of $\phi \in L^2$:

$$|\tilde{q} - \mathbb{E}_{\mathbf{x},y}[\phi(\mathbf{x}, y)]| \leq \tau.$$

- **Correlational statistical query (CSQ).** ϕ restricted to be *correlational*:

$$|\tilde{q} - \mathbb{E}_{\mathbf{x},y}[\phi(\mathbf{x})y]| \leq \tau.$$

❒ *Connection to sample complexity*: $\tau \approx n^{-1/2}$ $\Leftrightarrow$ i.i.d. concentration error.

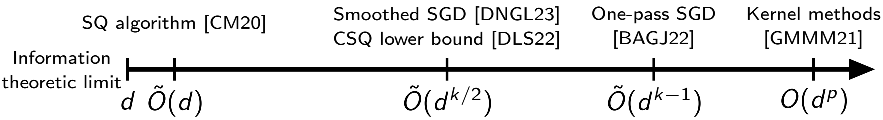
Theorem ([Damian et al. 22], [Abbe et al. 23], [Damian et al. 24]...)

*To learn polynomial f_* with information exponent k (using **polynomial compute**),*

- ***CSQ learner** requires $n \gtrsim d^{k/2}$ samples.*
- ***SQ learner** requires $n \gtrsim d$ samples.*

Remark: SQ learners may nonlinearly transform y to lower the information exponent.

Outline of This Talk



Complexity of learning single-index polynomial w. degree p & information exponent k.

❒ **Part 1:** SGD implements SQ and learns polynomial f_* in $n = \tilde{O}(d)$ samples

- By *reusing the same training examples* in the gradient computation, SGD implements nonlinear transformation that *lowers the information exponent*.

❒ **Part 2:** Learning sum of M *single-index models*, $M \asymp d^\gamma$ (*extensive rank*)

- Efficient gradient-based training of two-layer NNs.
- Computational hardness measured by (C)SQ lower bounds.

❒ **Part 3:** Learning *rank-r* single-index function class in-context via transformer

- Pretrained transformer achieves in-context complexity that only depends on the dimensionality of function class $r \ll d$.

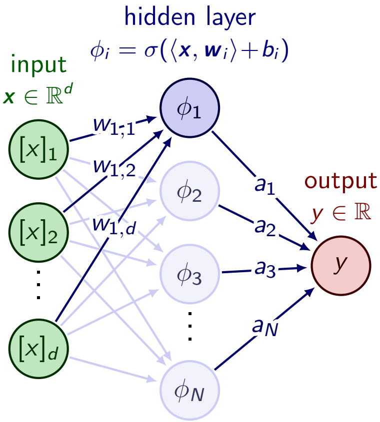
Width-N Two-layer NN: $f_{\mathrm{NN}}(\boldsymbol{x}) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} a_i \sigma(\langle \boldsymbol{x}, \boldsymbol{w}_i \rangle + b_i)$.

Architecture

- Randomized activation (with non-zero Hermite coefficient up to certain degree).
  - Required to establish strong recovery.
- Untrained random bias units.
  - Required to approximate unknown link.

Training Algorithm

- Layer-wise SGD training.
  - First-layer finds target direction $\boldsymbol{\theta}$, second-layer fits link function σ_*.
- Same data used in *two consecutive updates*.



Motivation: Can SGD Go Beyond CSQ?

Theorem ([Mondelli & Montanari 18], [Barbier et al. 19], [Chen & Meka 20]...)

For any polynomial σ_*, there exists $\mathcal{T}$ s.t. $\mathbb{E}[\mathcal{T}(\sigma_*(z))\mathrm{He}_i(z)] \neq 0$ *for* $\underline{i = 1 \text{ or } 2}$.

Question: can SGD with *squared loss* utilize such label transformations?

- **[Dandi et al. 24]** SGD + reused batch gives higher-order (non-correlational) info.
- **Intuition.** Consider two consecutive GD steps on $(\boldsymbol{x}, y)$, starting from $\boldsymbol{w}^{(0)} = 0$.

$$\boldsymbol{w}^{(2)} = \boldsymbol{w}^{(1)} + \eta \cdot y\sigma'(\langle \boldsymbol{x}, \boldsymbol{w}^{(1)} \rangle)\boldsymbol{x} = \eta\sigma'(0) \underbrace{y \cdot \boldsymbol{x}}_{\text{CSQ term}} + \eta \underbrace{y\sigma'(\eta\sigma'(0)\|\boldsymbol{x}\|^2 \cdot y)\boldsymbol{x}}_{\text{non-CSQ term}}.$$

Can NN optimized by SGD + reused batch learn *arbitrary* single-index polynomials near the information-theoretic limit $n \asymp d$, regardless of the information exponent?

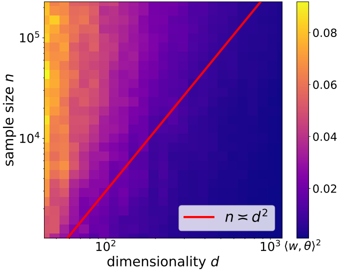
Motivation: Can SGD Go Beyond CSQ?

Theorem ([Mondelli & Montanari 18], [Barbier et al. 19], [Chen & Meka 20]...)

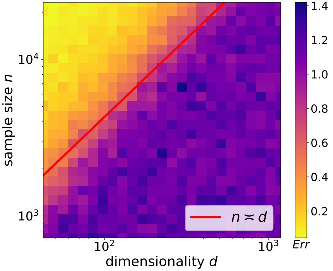
For any polynomial σ_*, there exists $\mathcal{T}$ s.t. $\mathbb{E}[\mathcal{T}(\sigma_*(z))\mathrm{He}_i(z)] \neq 0$ *for* $\underline{i = 1 \text{ or } 2}$.

Question: can SGD with *squared loss* utilize such label transformations?

Empirically: Yes! $f_*(\boldsymbol{x}) = \mathrm{He}_3(\langle \boldsymbol{x}, \boldsymbol{\theta}\rangle)$, $f_{\mathrm{NN}}(\boldsymbol{x}) = \sum_{i=1}^N a_i \mathrm{ReLU}(\langle \boldsymbol{x}, \boldsymbol{w}_i\rangle + b_i)$.



(a) Online SGD (weak recovery)



(b) Same-batch GD (test error)

SGD Training of Two-layer Neural Network

Algorithm 1: Gradient-based training of two-layer neural network

Input : Learning rates η^t, momentum parameter ξ^t, number of steps T_1, T_2, ℓ_2 regularization λ.

Initialize $\boldsymbol{w}_j^0 \sim \mathrm{Unif}(\mathbb{S}^{d-1}(1))$, $a_j \sim \mathrm{Unif}\{\pm r_a\}$.

Phase I: normalized SGD on first-layer parameters

for $t = 0$ ***to*** T_1 **do**

- $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$, $y = f_*(\boldsymbol{x}) + \varsigma$; // Draw i.i.d. training example $(\boldsymbol{x}, y)$
- $\tilde{\boldsymbol{w}}_j^t \leftarrow \boldsymbol{w}_j^t - \eta^t \tilde{\nabla}_{\boldsymbol{w}} (f_{\Theta_t}(\boldsymbol{x}) - y)^2$; // First gradient descent step
- $\tilde{\boldsymbol{w}}_j^t \leftarrow \tilde{\boldsymbol{w}}_j^t - \eta^t \tilde{\nabla}_{\boldsymbol{w}} (f_{\tilde{\Theta}_t}(\boldsymbol{x}) - y)^2$; // Second gradient descent step
- $\boldsymbol{w}_j^{t+1} \leftarrow \tilde{\boldsymbol{w}}_j^t - \xi^t(\tilde{\boldsymbol{w}}_j^t - \boldsymbol{w}_j^t)$; // Interpolation step
- $\boldsymbol{w}_j^{t+1} \leftarrow \boldsymbol{w}_j^{t+1} / \|\boldsymbol{w}_j^{t+1}\|$, $(j = 1, \ldots, N)$; // Normalization

end

Initialize $b_j \sim \mathrm{Unif}([-C_b, C_b])$.

Phase II: SGD on second-layer parameters

- $\hat{\boldsymbol{a}} \leftarrow \mathrm{argmin}_{\boldsymbol{a} \in \mathbb{R}^N} \frac{1}{T_2} \sum_{i=1}^{T_2} (f_\Theta(\boldsymbol{x}_i) - y_i)^2 + \lambda \|\boldsymbol{a}\|^2$; // Ridge regression estimator

Output: Prediction function $\boldsymbol{x} \mapsto f_{\hat{\Theta}}(\boldsymbol{x})$ with $\hat{\Theta} = (\hat{a}_j, \boldsymbol{w}_j^{T_1}, b_j)_{j=1}^N$.

- **Ingredient I:** resample batch in *every two steps*.
- **Ingredient II:** interpolation & normalization to stabilize dynamics.

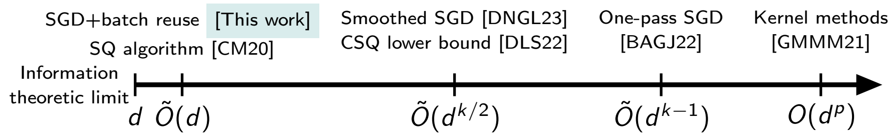
SGD is Almost Information Theoretically Optimal

Theorem ([LOSW24] Complexity of SGD Training)

For arbitrary single-index polynomial target functions, Algorithm 1 (w. appropriate hyperparameters) achieves population loss $\mathbb{E}_{\boldsymbol{x}}[(f_*(\boldsymbol{x}) - f_{\mathrm{NN}}(\boldsymbol{x}))^2] \leq \varepsilon$ *using*

$$n = \tilde{O}_d(d\varepsilon^{-2}), \qquad N = \tilde{O}_d(\varepsilon^{-1}).$$

- Algorithm almost agnostic to link function (only requires knowledge of *degree* p).
- Hides constant C_p that depends *exponentially* on the degree p.



Complexity of learning single-index polynomial w. degree p & information exponent k.

Key Ingredients in the Analysis

Ingredient I: *Polynomial transformation lowers information exponent*

Proposition ([LOSW24] Existence of monomial transformation)

- *If* σ_* *is even, there exists* $i \le C_p \in \mathbb{N}_+$ *such that* $\mathrm{IE}(\sigma_*^i) = 2$,
- *If* σ_* *is not even, there exists* $i \le C_p \in \mathbb{N}_+$ *such that* $\mathrm{IE}(\sigma_*^i) = 1$,

for some uniform upper bound C_p *depending only on the degree* p.

Ingredient II: *SGD with reused batch implements monomial transformation*

$$\sigma_*(z) = \textstyle\sum_{i=0}^{p} \alpha_i^* \mathrm{He}_i(z), \quad \sigma(z) = \textstyle\sum_{i=0}^{C_p} \beta_i \mathrm{He}_i(z).$$

- For weak recovery, we need $\mathbb{E}[\mathrm{He}_j(z)\sigma^{(i)}(z)(\sigma^{(1)}(z))^{i-1}] \neq 0$, for $i \le C_p$, $j = 0, 1$.
- For strong recovery, Hermite coefficients should satisfy $\alpha_j \beta_j \ge 0$ for $k \le j \le p$.

Remark: both conditions satisfied when β_i are randomly drawn, w.p. $\Omega(1)$.

Beyond Polynomial Link Functions

Question: Can we go beyond learning single-index *polynomials*?

Definition: generative exponent [Damian et al. 2024]

The generative exponent of σ_* is defined as $k_* := \min_{\mathcal{T} \in L^2(\gamma)} \mathrm{IE}(\mathcal{T} \circ \sigma_*)$.

Interpretation: smallest information exponent after *arbitrary L^2 transformation*.

- For any *polynomial* σ_*, $k_* \leq 2$.
- For $\sigma_*(z) = z^2 \exp(-z^2)$, $k_* = 4$.

Theorem ([LOSW24] SGD for Higher Generative Exponent σ_*)

For arbitrary single-index models with generative exponent k_* and $\sigma_*, \sigma_*'' \in L^4(\gamma)$, Algorithm 1 achieves population loss $\mathbb{E}_{\boldsymbol{x}}[(f_*(\boldsymbol{x}) - f_{\mathrm{NN}}(\boldsymbol{x}))^2] \leq o_{d,\mathbb{P}}(1)$ using

$$n \simeq T \gg \begin{cases} d & (\text{if } k_* = 1) \\ d \log d & (\text{if } k_* = 2) \\ d^{p_* - 1} & (\text{if } k_* \geq 3). \end{cases}$$

Motivation: Learning Diverse Features Simultaneously

Additive Model with M Tasks (ridge combinations)

$$f_*(\boldsymbol{x}) = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} \sigma_m(\langle \boldsymbol{x}, \boldsymbol{\theta}_m \rangle), \qquad M \asymp d^\gamma \text{ for } \gamma > 0 .$$

- **Link functions:** $\sigma_m : \mathbb{R} \to \mathbb{R}$ has degree p and information exponent k.
- **Diversity of tasks:** $M \lesssim \left(\max_{m \neq m'} \langle \boldsymbol{\theta}_m, \boldsymbol{\theta}_{m'} \rangle^2 \right)^{-1/2} \wedge d^{1/2}$.

  $\Rightarrow$ e.g., $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_M \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}(\mathbb{S}^{d-1}(1))$ with $M \lesssim d^{1/2}$.

- **Question 1.** Can we learn f_* via gradient-based training of two-layer neural network? What is the *statistical and computational complexity* of SGD?
- **Question 2.** What is the *computational hardness* of learning this additive model class, and how does it differ from the previously studied finite-M setting?

Efficiency of Online SGD Learning

Theorem ([OSSW24] Statistical Complexity of SGD Training)

For $k > 2$, layer-wise (online) SGD training of two-layer neural network achieves ε population loss using

$$n = \tilde{O}_d\left(Md^{k-1} \vee Md\varepsilon^{-2}\right), \qquad N = \tilde{O}_d\left(M^{C_k+1/2}\varepsilon^{-1}\right),$$

where constant $C_k = \max_{m \neq m'} \left|\alpha_k^m / \alpha_k^{m'}\right| \geq 1$.

Comparison against prior results.

- **Kernel ridge regression** requires $n \gtrsim d^p$ samples.
  - KRR does not adapt to low-dimensional structures.
- **GD-based training for multi-index model** requires $n \gtrsim \left(M^p \vee d^{\Theta(k)}\right)$ samples.
  - Does not take into account the *additive structure* of f_*
    $\Rightarrow$ statistical complexity worsen when M becomes large.

Prior analysis: ***subspace random features*** [Damian et al. 22], [Abbe et al. 23],...

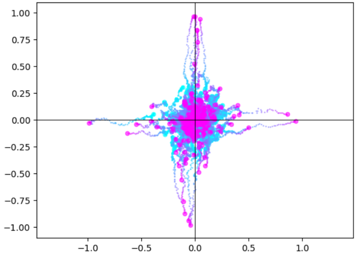
Gradient-based feature learning "localizes" parameters into *rank-M subspace*.

Our analysis: ***task localization***

After first-layer training, for each task $\boldsymbol{\theta}_m$, there exists some student neurons $\boldsymbol{w}_j$ s.t.

$$\langle \boldsymbol{\theta}_m, \boldsymbol{w}_j \rangle \geq 1 - \varepsilon.$$

- **Fine-tuning:** if downstream task consists of $\tilde{M} \ll M$ directions, $n \gtrsim \tilde{M}\varepsilon^{-2}$ samples needed.



Statistical Query Lower Bounds

Heuristic: we equate the tolerance with the scale of concentration error $\tau \approx n^{-1/2}$

Theorem ([OSSW24] CSQ Lower Bound)

For a CSQ algorithm to learn f_* using polynomially many queries, we must have

$$n \gtrsim M \cdot d^{k/2}$$

For CSQ, learning additive model with M tasks $\approx$ learning M single-index models.

Theorem ([OSSW24] SQ Lower Bound)

*Given fixed $M \asymp d^\gamma$ with $\gamma > 0$, for **any** $\rho > 0$, there exists some σ_* with degree p depending only on ρ, γ, such that an SQ learner (with polynomial compute) requires*

$$n \gtrsim (M \cdot d)^\rho$$

For SQ, learning additive model with M tasks $\neq$ learning M single-index models.

SQ Lower Bound Derivation (sketch)

Proposition ([OSSW24] "Superorthogonal" Polynomials)

For any $K, I \in \mathbb{N}_+$, there exists a non-zero polynomial $g : \mathbb{R} \to \mathbb{R}$ that satisfies:

$$\mathbb{E}_z[(g(z))^i \mathrm{He}_k(z)] = 0,$$

for every $1 \le k \le K$ and $1 \le i \le I$.

Intuition: given fixed $I \in \mathbb{N}_+$, there exist *polynomial* link functions such that polynomial transformations up to degree I cannot lower its information exponent.

(i) For $I = 1$ and $K \in \mathbb{N}$, $g(z) = \mathrm{He}_{K+1}(z)$.

(ii) For $I = K = 2$, $g(z) = \mathrm{He}_4(z) - \frac{4}{15}\mathrm{He}_6(z) + \frac{11}{280}\mathrm{He}_8(z) - \frac{19}{4725}\mathrm{He}_{10}(z) + \frac{311}{997920}\mathrm{He}_{12}(z) - \frac{719}{37837800}\mathrm{He}_{14}(z) + \frac{14297}{15567552000}\mathrm{He}_{16}(z) - \frac{35369}{1042053012000}\mathrm{He}_{18}(z) + \left(\frac{35369}{41682120480000} - \frac{1}{83364240960000}\sqrt{\frac{11163552839}{38}}\right)\mathrm{He}_{20}(z)$.

❒ Why is restriction to **fixed-degree** polynomial transformations sufficient?

- When $M \to \infty$, the statistical query $\phi(\boldsymbol{x}, y)$ applied to one single-index task can be *Taylor expanded*, which limits the available transformations.

Complexity of Learning Additive Models

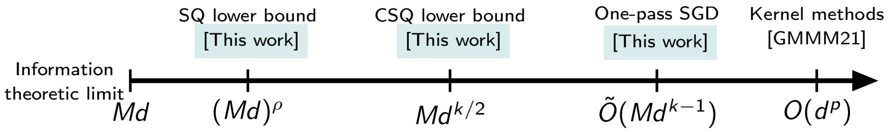


Figure 1: Complexity of learning width-M additive model w. degree p & information exponent k.

❐ Computational-statistical gap

- Learning is information-theoretically possible with $n \gtrsim Md$ samples.
- SQ learner requires $n \gtrsim (Md)^{\rho}$ where ρ can be made *arbitrarily large*.

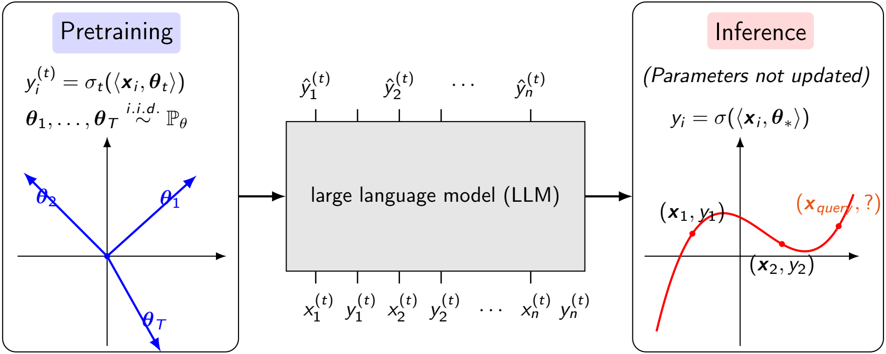
❐ Closing the sample complexity gap

- Match CSQ rate via a smoothing procedure?
- Match SQ rate via reusing batch?

Motivation: Learning Single-index Models In-Context

In-context learning [Brown et al. 2020]

Observation: LLMs can learn *in-context*, i.e., construct new predictors from labeled examples (context) presented in the input *without parameter updates*.



Intuition: LLM can implement (efficient) algorithms in its *forward pass*.

Motivation: Why Single-index Models?

Prior Results: pretrained *linear* transformer (TF) learns *linear* functions in context.

Theorem ([Zhang et al. 23], [Ahn et al. 23], [Mahankali et al. 23],...)

Linear TF pretrained on linear function class $\mathcal{F}_{\mathrm{lin}} = \{f \mid f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{\theta} \rangle, \boldsymbol{\theta} \sim \mathbb{S}^{d-1}(1)\}$ *achieves in-context (roughly) prediction risk competitive with the* ***best linear model****.*

- Expressivity beyond linear models?
  - Linear TF can implement limited algorithms, e.g., *linear regression*.
  - Single-index model is a natural nonlinear generalization of linear predictor.

- Adaptivity to structure of function class?
  - Solving single-index regression on test prompt requires *long context*.
    $\Rightarrow$ kernel: $n \gtrsim d^p$. CSQ: $n \gtrsim d^{\Theta(k)}$. SQ: $n \gtrsim d$.
  - TF should *adapt to target function class* via pretraining.
    $\Rightarrow$ improved ICL efficiency (e.g., ridge vs. LASSO [Garg et al. 22]).

Adaptivity to Low-dimensional Function Class

Definition (**Gaussian single-index model on rank-r subspace**)

Define the function class $\mathcal{F}_r^{k,p}$ in which $f(\boldsymbol{x}) = \sigma(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle)$, $\boldsymbol{x} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \boldsymbol{I}_d)$, and

- ❒ $\sigma : \mathbb{R} \to \mathbb{R}$ has degree at most p and information exponent at least k.
- ❒ $\boldsymbol{\theta}$ is drawn uniformly from fixed rank-r subspace where $r \ll d$, $\|\boldsymbol{\theta}\| = 1$.

Number of in-context examples n required to learn $f \in \mathcal{F}_r^{k,p}$

☹ For algorithms that directly learn f from the test prompt, $n \gtrsim d$ necessary.

- Kernel method: $n \gtrsim d^p$. CSQ algorithm: $n \gtrsim d^{\Theta(k)}$. SQ algorithm: $n \gtrsim d$.

☺ For algorithms that *find rank-r subspace* via pretraining, $n \gtrsim \mathrm{poly}(r)$ sufficient.

Can a pretrained TF learn the single-index function class $\mathcal{F}_r^{k,p}$ with an in-context sample complexity *independent of the ambient dimensionality d*?

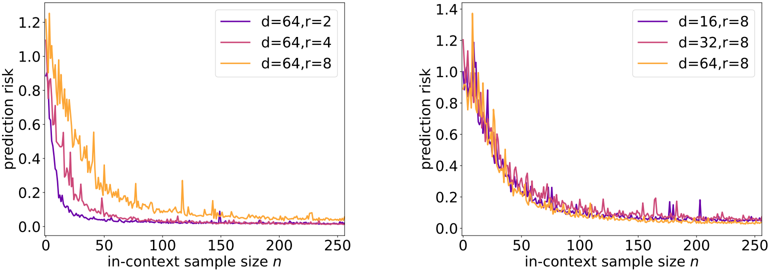
Adaptivity to Low-dimensional Function Class

Definition (Gaussian single-index model on rank-r subspace)

Define the function class $\mathcal{F}_r^{k,p}$ in which $f(\boldsymbol{x}) = \sigma(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle)$, $\boldsymbol{x} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \boldsymbol{I}_d)$, and

❒ $\sigma : \mathbb{R} \to \mathbb{R}$ has degree at most p and information exponent at least k.

❒ $\boldsymbol{\theta}$ is drawn uniformly from fixed rank-r subspace where $r \ll d$, $\|\boldsymbol{\theta}\| = 1$.



- 12-layer GPT2 model (~22M parameters) + Adam used in [Garg et al. 22].

Gradient-based Training of Attention Model

Linear Attention Module with MLP Layer

$$f_{\mathrm{Attn}}(\boldsymbol{E}; \boldsymbol{W}^{PV}, \boldsymbol{W}^{KQ}) = \boldsymbol{E} + \boldsymbol{W}^{PV}\boldsymbol{E} \cdot \left(\frac{\boldsymbol{E}^{\top}\boldsymbol{W}^{KQ}\boldsymbol{E}}{\rho} \right)$$

where

$$\boldsymbol{E} = \begin{bmatrix} \sigma(\boldsymbol{w}_1^{\top}\boldsymbol{x}_1 + b_1) & \cdots & \sigma(\boldsymbol{w}_1^{\top}\boldsymbol{x}_n + b_1) & \sigma(\boldsymbol{w}_1^{\top}\boldsymbol{x}_{query} + b_1) \\ \vdots & \ddots & \vdots & \vdots \\ \sigma(\boldsymbol{w}_N^{\top}\boldsymbol{x}_1 + b_N) & \cdots & \sigma(\boldsymbol{w}_N^{\top}\boldsymbol{x}_n + b_N) & \sigma(\boldsymbol{w}_N^{\top}\boldsymbol{x}_{query} + b_N) \\ y_1 & \cdots & y_n & 0 \end{bmatrix}$$

- Trainable MLP (embedding) weights $\boldsymbol{W}$ to adapt to low-dimensional structure.
- Nonlinear activation $\sigma = \mathrm{ReLU}$ to express nonlinear labels.

Alternatively, we can introduce the reparameterization $\boldsymbol{\Gamma} \in \mathbb{R}^{N\times N}$ and write

$$f(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_{query}; \boldsymbol{W}, \boldsymbol{\Gamma}, \boldsymbol{b}) = \left\langle \frac{1}{N}\boldsymbol{\Gamma}\sigma\left(\boldsymbol{W}\boldsymbol{X} + \boldsymbol{b}1_N^{\top}\right)\boldsymbol{y}, \; \sigma\left(\boldsymbol{W}^{T}\boldsymbol{x}_{query} + \boldsymbol{b}\right) \right\rangle.$$

Gradient-based Training of Transformer

Algorithm 2: Gradient-based training of transformer with MLP layer

Input : Learning rate η_1, weight decay λ_1, λ_2, prompt length n_1, n_2, number of tasks T_1, T_2.

Initialize $\boldsymbol{w}_j^{(0)} \sim \mathrm{Unif}(\mathbb{S}^{d-1})$ $(j \in [m])$; $b_j^{(0)} \sim \mathrm{Unif}([-1, 1])$ $(j \in [m])$;
$\Gamma_{j,j}^{(0)} \sim \mathrm{Unif}(\{\pm\gamma\})$ $(j \in [m])$ *and* $\Gamma_{i,j}^{(0)} = 0$ $(i \neq j \in [m])$.

Phase I: Gradient descent for MLP layer

Draw data $\{(\boldsymbol{x}_1^t, y_1^t, \ldots, \boldsymbol{x}_{n_1}^t, y_{n_1}^t, \boldsymbol{x}^t, y^t)\}_{t=1}^{T_1}$ with prompt length n_1.

$\boldsymbol{w}_j^{(1)} \leftarrow \boldsymbol{w}_j^{(0)} - \eta_1 \left[\nabla_{\boldsymbol{w}_j} \frac{1}{T_1} \sum_{t=1}^{T_1} (y^t - f(\boldsymbol{W}^{(0)}, \boldsymbol{\Gamma}^{(0)}, \boldsymbol{b}^{(0)}))^2 + \lambda_1 \boldsymbol{w}_j^{(0)}\right]$; // one GD step

Initialize $b_j \sim \mathrm{Unif}([-C_b \log d, C_b \log d])$.

Phase II: Empirical risk minimization for attention layer

Draw data $\{(\boldsymbol{x}_1^t, y_1^t, \ldots, \boldsymbol{x}_{N_2}^t, y_{n_2}^t, \boldsymbol{x}^t, y^t)\}_{t=T_1+1}^{T_1+T_2}$ with prompt length n_2.

$\boldsymbol{\Gamma}^* \leftarrow \mathrm{argmin}_{\boldsymbol{\Gamma}} \frac{1}{T_2} \sum_{t=T_1+1}^{T_1+T_2} (y^t - f(\boldsymbol{W}^{(1)}, \boldsymbol{\Gamma}, \boldsymbol{b}))^2 + \frac{\lambda_2}{2} \|\boldsymbol{\Gamma}\|_F^2$; // ridge regression

Output: trained parameters $(\boldsymbol{W}^{(1)}, \boldsymbol{\Gamma}^*, \boldsymbol{b})$.

- **Ingredient I:** *one GD step* on MLP layer to identify rank-r subspace.
  - Gradient of correlation term spans r-dimensional subspace [Damian et al. 22].
- **Ingredient II:** train *attention layer* to approximate nonlinear link function.
  - Attention layer performs regression on polynomial basis defined by MLP layer.

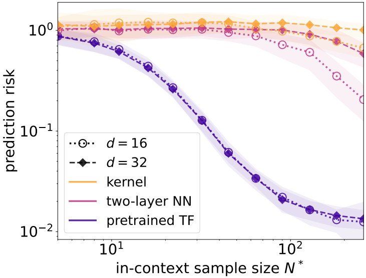
Dimension-free In-context Sample Complexity

Theorem ([OSSW24] Sample Complexity of ICL)

TF trained by Algorithm 2 achieves prediction risk $\mathbb{E}|f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{\Gamma}, \boldsymbol{b}) - f_*(\boldsymbol{x})| = o_d(1)$*, with high probability, if the number of pretraining tasks* T*, the number of training examples* n*, the test prompt length* n^**, and the number of neurons* N *satisfy*

$$\underbrace{n, T \gtrsim d^{\Theta(k)}}_{\text{pretraining cost}}, \quad \underbrace{n^* \gtrsim r^{\Theta(p)}}_{\text{inference cost}}, \quad \underbrace{N \gtrsim r^{\Theta(p)}}_{\text{approximation}}.$$

- **Pretraining** sample complexity scales with *ambient dimensionality* d.
- **In-context** sample complexity scales with the *target subspace dimensionality* $r < d$.
  - **Adaptivity:** in-context complexity parallel to *r-dimensional polynomial regression.*



References

Thank you! Happy to take questions :)

- Vaswani et al., 2017. *Attention is all you need*.
- Ghorbani et al., 2020. *Linearized two-layers neural networks in high dimension*.
- Chen and Meka, 2020. *Learning polynomials of few relevant dimensions*.
- Brown et al., 2020. *Language models are few-shot learners*.
- Ben Arous et al., 2021. *Stochastic gradient descent on non-convex losses from high-dimensional inference*.
- Bietti et al., 2022. *Learning single-index models with shallow neural networks*.
- Garg et al., 2022. *What can transformers learn in-context? A case study of simple function classes*.
- Damian et al., 2023. *Smoothing the landscape boosts the signal for SGD: optimal sample complexity for learning single index models*.
- Dandi et al., 2024. *The benefits of reusing batches for gradient descent in two-layer networks: breaking the curse of information and leap exponents*.
- Damian et al., 2024. *Computational complexity of learning Gaussian single-index models*.