Over-Parametrized Neural Networks and Sobolev Training

IMSI Statistical and Computational Challenges in Probabilistic Scientific Machine Learning

2025-06-12

Matt Li (mtcli@mit.edu / mtcli@umass.edu)

This is **ongoing work** with my ~~large language models~~ collaborators

Kate Fisher (MIT)

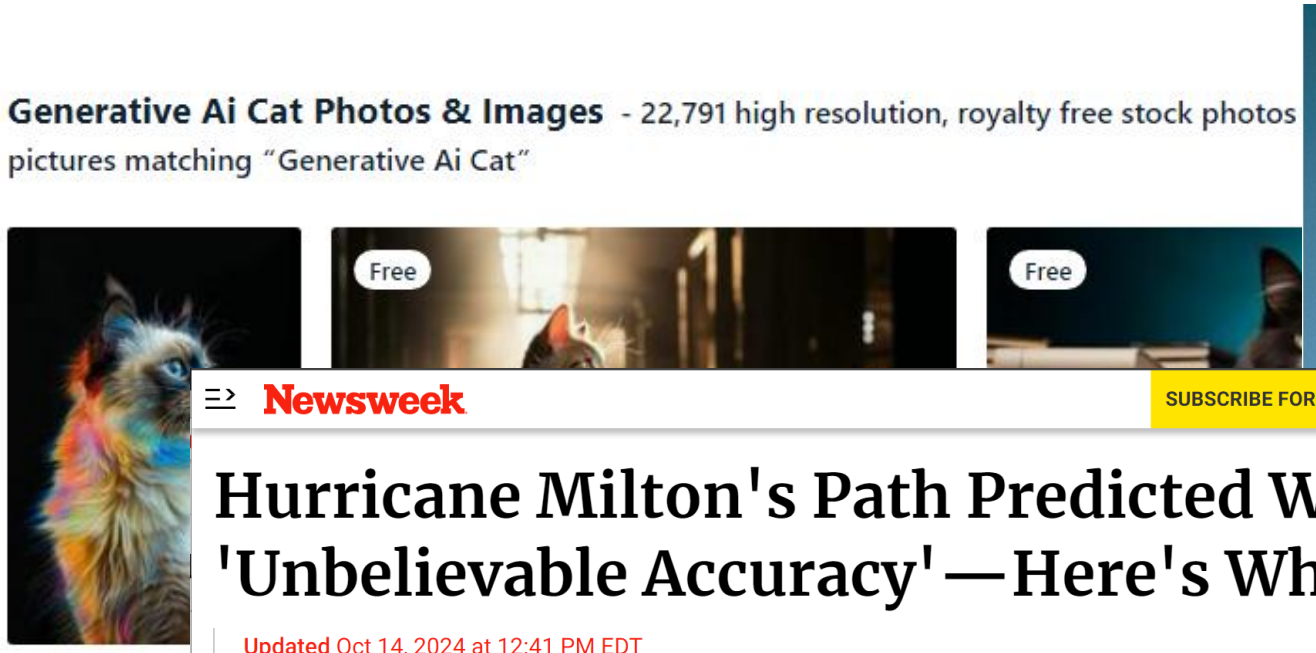
Timo Schorlepp (NYU)

Youssef Marzouk (MIT)

Supported by the US Department of Energy, Office of Advanced Scientific Computing Research, under grant DE-SC0023187 (M2DT: Multifaceted Mathematics for Predictive Digital Twins.)

The era of **scientific machine learning**

Machine learning algorithms have reshaped how we approach statistical inference.



Newsweek SUBSCRIBE FOR $1 Login

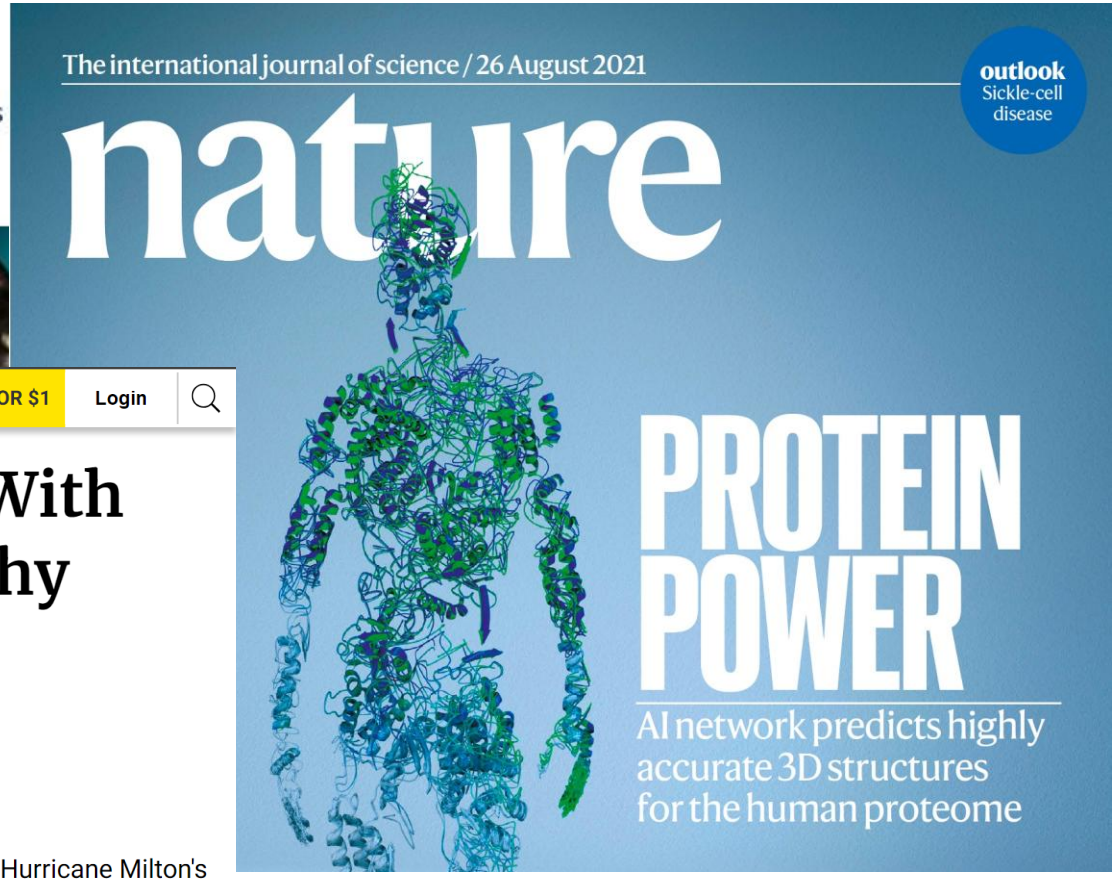
Hurricane Milton's Path Predicted With 'Unbelievable Accuracy'—Here's Why

Updated Oct 14, 2024 at 12:41 PM EDT

By Marie Boran
Technology Reporter

FOLLOW

A weather model driven by artificial intelligence has stunned meteorologists by predicting Hurricane Milton's landfall within 7 miles on average—outperforming traditional models by almost 100 miles.



Big data and even bigger models

nature › news feature › article

NEWS FEATURE | 11 December 2024

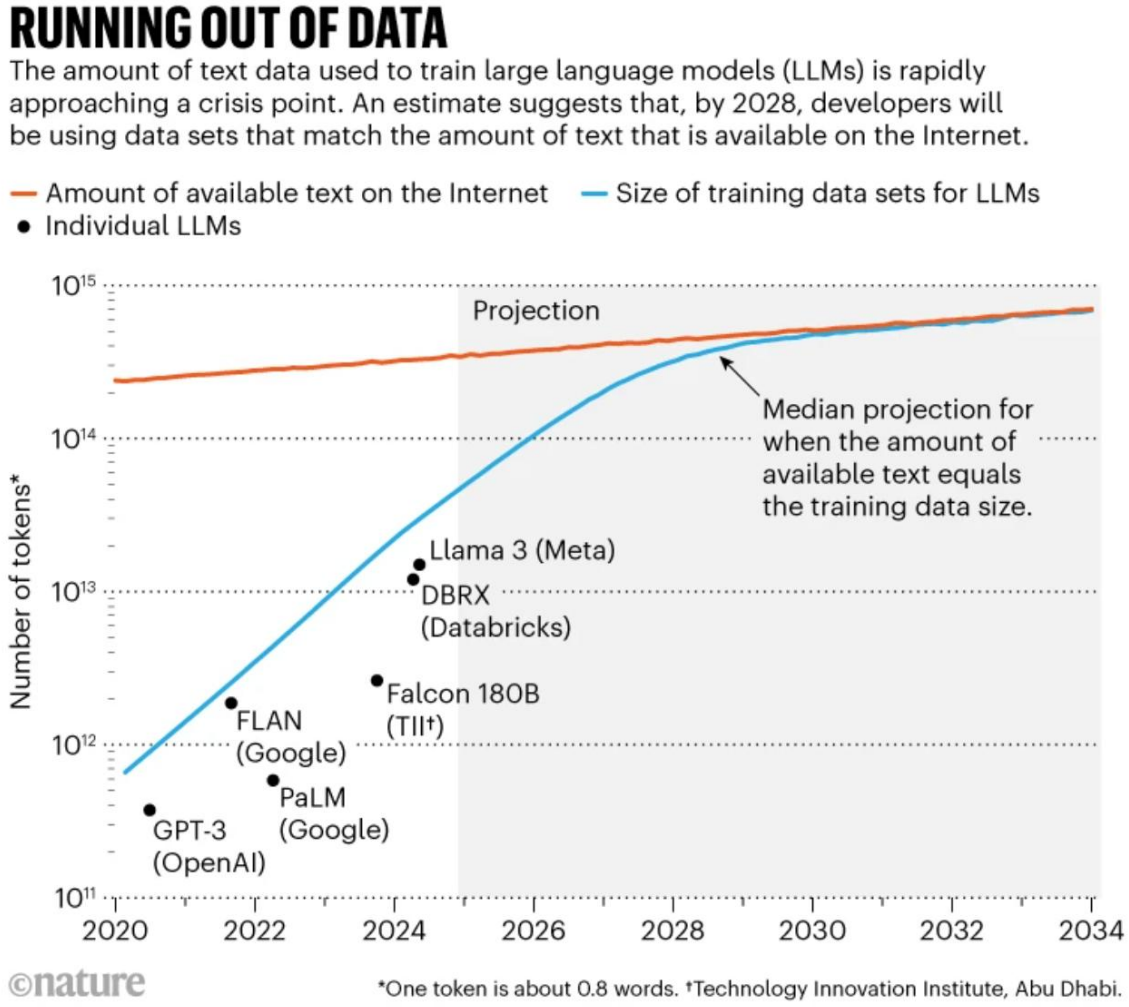
The AI revolution is running out of data. What can researchers do?

AI developers are rapidly picking the Internet clean to train large language models such as those behind ChatGPT. Here's how they are trying to get around the problem.

By Nicola Jones

BENZINGA Research My Stocks Tools

OpenAI Co-Founder Ilya Sutskever Rings Alarm Bells: AI's 'Fossil Fuel' Is Running Out As World Reaches 'Peak Data'

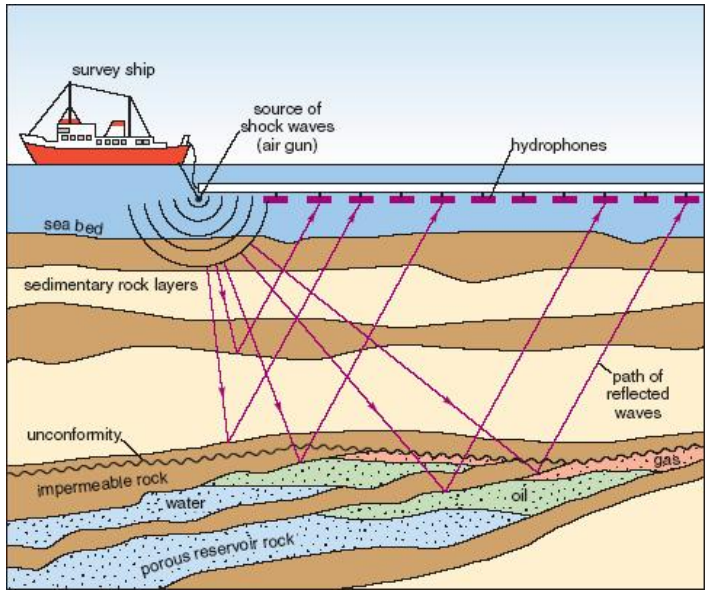


Scientific machine learning is **data bottlenecked**

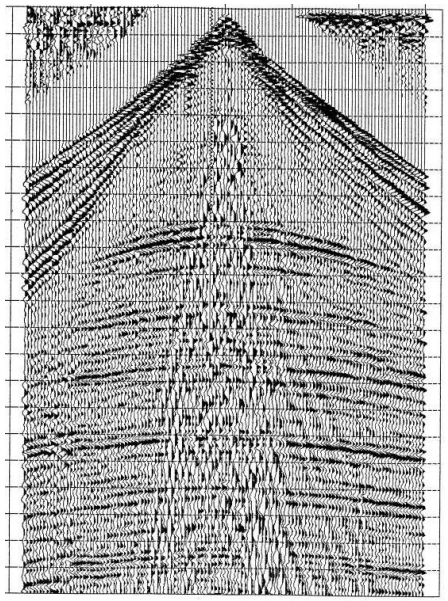
Experimental or numerical data are extremely scarce in scientific applications!!!

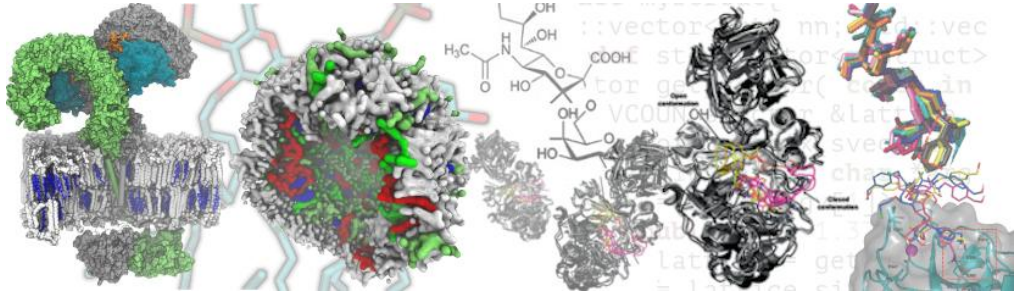
The prevailing wisdom is to **increase data efficiency by incorporating physical priors**, e.g., by leveraging symmetries, invariances, and equivariances, or applying data augmention.

Model Parameters x



Data y





Derivatives are the language of science and engineering

What happens if we also provide information about derivatives?

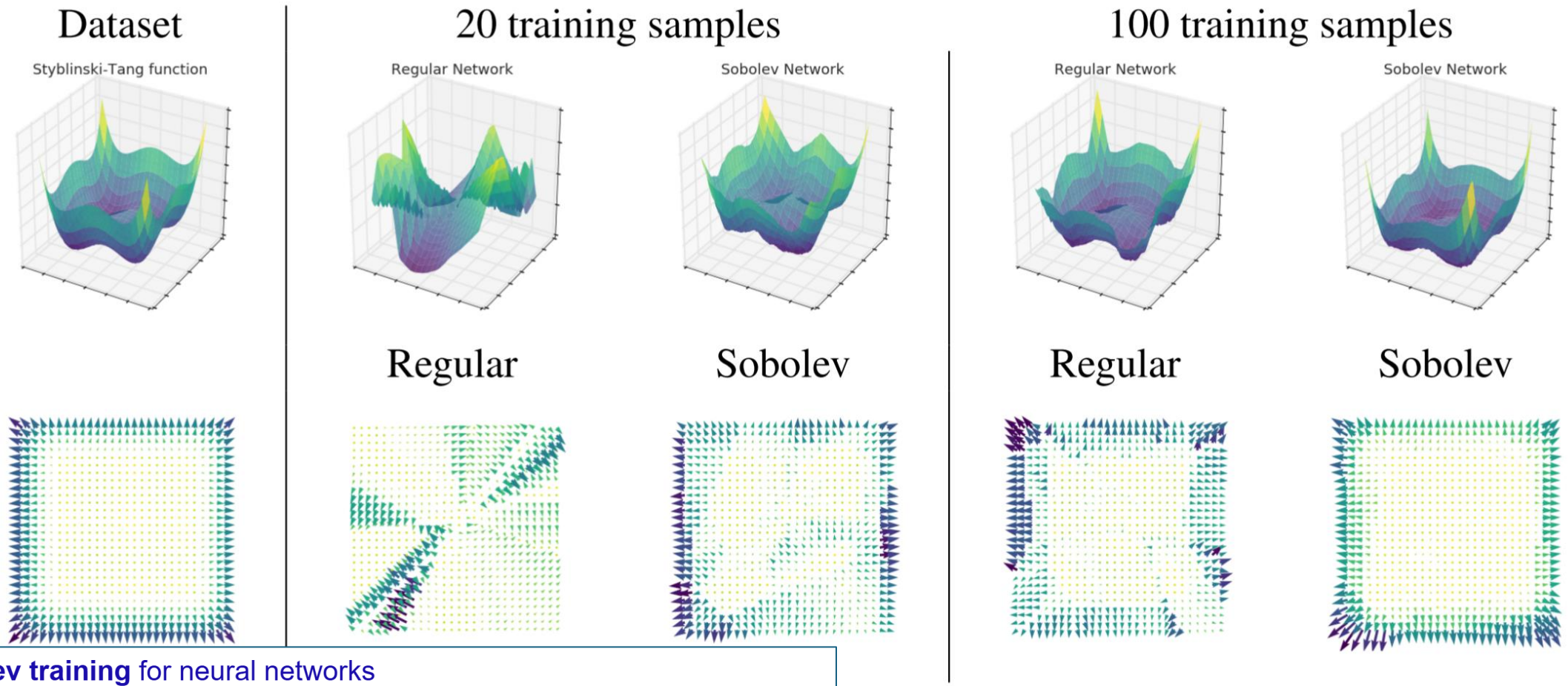
Given training data $\{x_i, y_i, y_i'\}_{i\in[n]}$ minimize the empirical risk

$$\min_{f_{\mathsf{NN}}:\mathbb{R}^d\to\mathbb{R}} \frac{1}{n}\sum_{i=1}^{n}\left\{(y_i - f_{\mathsf{NN}}(x_i))^2 + \|y_i' - \nabla f_{\mathsf{NN}}(x_i)\|^2\right\} + \lambda\cdot \mathrm{reg}(f_{\mathsf{NN}})\ .$$

- **emulators for molecular dynamics** [Behler 2016]
- **derivative informed neural operators** [O'Leary-Roseberry et al 2024]
- **predicting summary statistics for chaotic dynamical systems** [Park et al 2025]
- **teacher student knowledge distillation** [Czarnecki et al 2017]

Behler, Jörg. "Perspective: Machine Learning Potentials for Atomistic Simulations." *The Journal of Chemical Physics*.
O'Leary-Roseberry, Thomas, Peng Chen, Umberto Villa, and Omar Ghattas. "Derivative-Informed Neural Operator: An Efficient Framework for High-Dimensional Parametric Derivative Learning." *Journal of Computational Physics*.
Park, Jeongjin, Nicole Yang, and Nisha Chandramoorthy. "When Are Dynamical Systems Learned from Time Series Data Statistically Accurate?"

Sobolev Training



Sobolev training for neural networks

WM Czarnecki, S Osindero… - Advances in neural …, 2017 - proceedings.neurips.cc

… In our method, we show that by using the additional knowledge of derivatives with **Sobolev Training** we are able to **train** better models – models which achieve lower approximation …

☆ Save 99 Cite Cited by 284 Related articles All 6 versions »

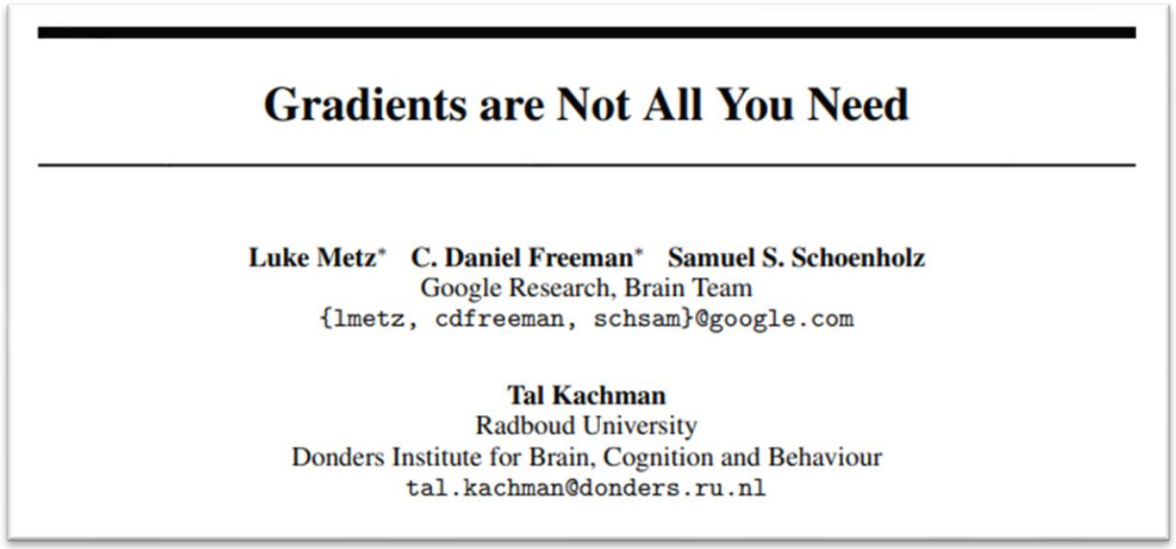
But see also "Hermite interpolation" in numerical analysis, "Hermite learning" with RKHS, etc...

But do derivative data actually **help or hurt** neural networks?

Question. Should we always incorporate derivative data if you have it? Should we always expect to get better neural networks?

But do derivative data actually **help or hurt** neural networks?

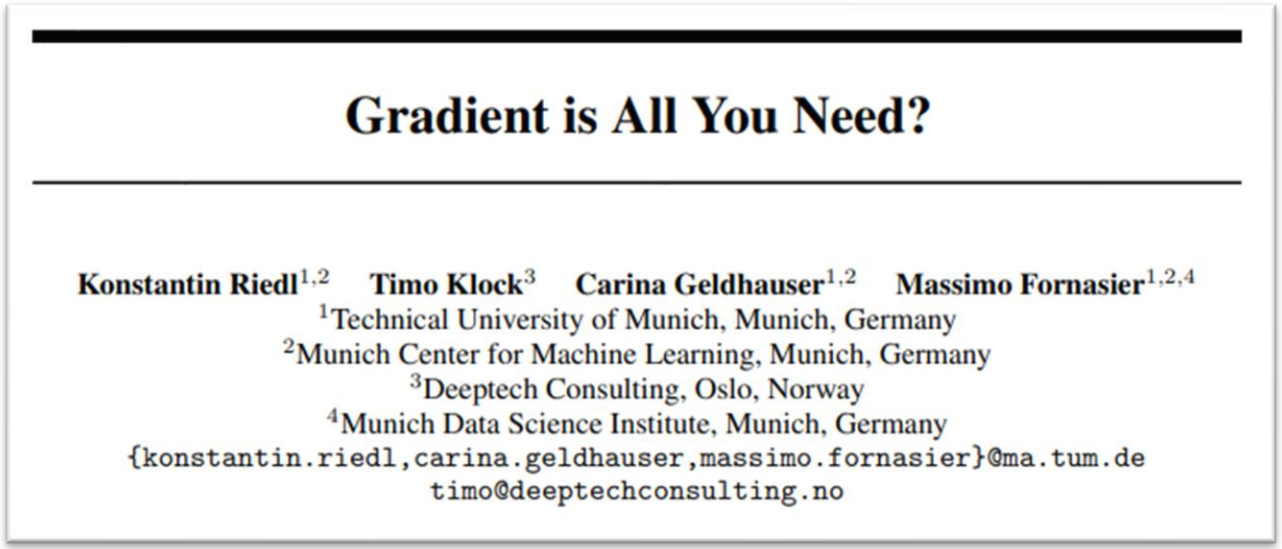
Question. Should we always incorporate derivative data if you have it? Should we always expect to get better neural networks?



Gradients are Not All You Need

Luke Metz* **C. Daniel Freeman*** **Samuel S. Schoenholz**
Google Research, Brain Team
{lmetz, cdfreeman, schsam}@google.com

Tal Kachman
Radboud University
Donders Institute for Brain, Cognition and Behaviour
tal.kachman@donders.ru.nl



Gradient is All You Need?

Konstantin Riedl[1,2] **Timo Klock**[3] **Carina Geldhauser**[1,2] **Massimo Fornasier**[1,2,4]
[1]Technical University of Munich, Munich, Germany
[2]Munich Center for Machine Learning, Munich, Germany
[3]Deeptech Consulting, Oslo, Norway
[4]Munich Data Science Institute, Munich, Germany
{konstantin.riedl,carina.geldhauser,massimo.fornasier}@ma.tum.de
timo@deeptechconsulting.no

FYI these papers have nothing to do with Sobolev training.

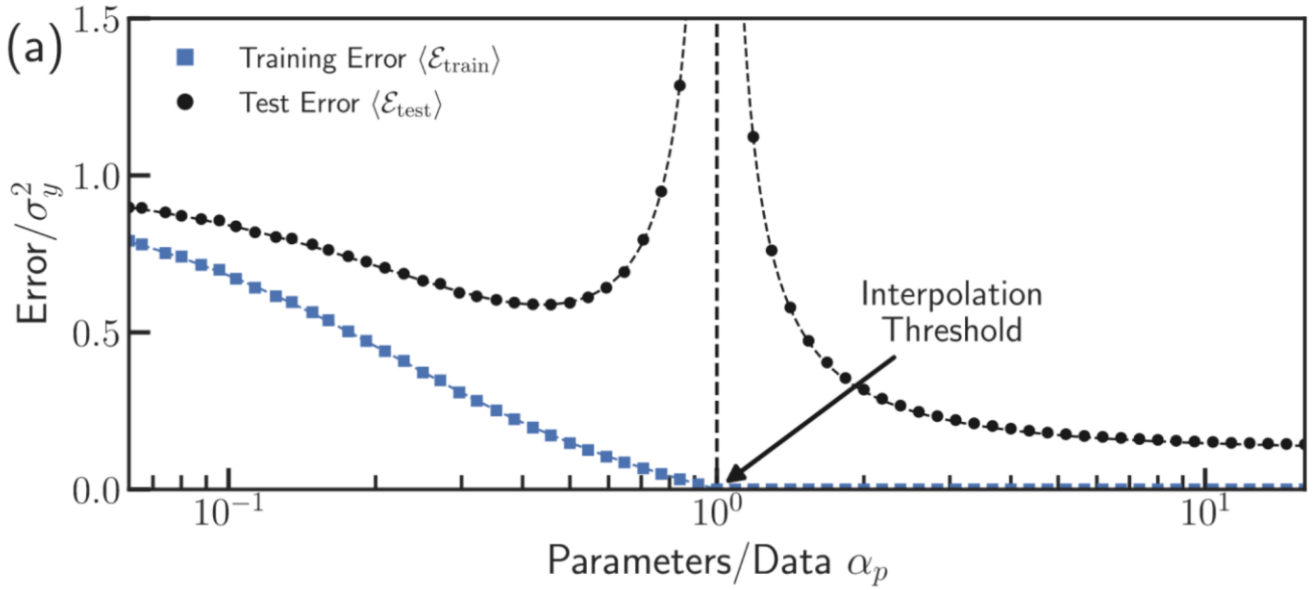
Perhaps papers claiming gradients are or aren't all you need . . . aren't all you need?

What is the role of **over-parametrization?**

In my opinion, machine learning is distinguished by the fact that

degrees of freedom(big models) $\gg$ # of big data.

We are able to interpolate data, yet still achieve good generalization.



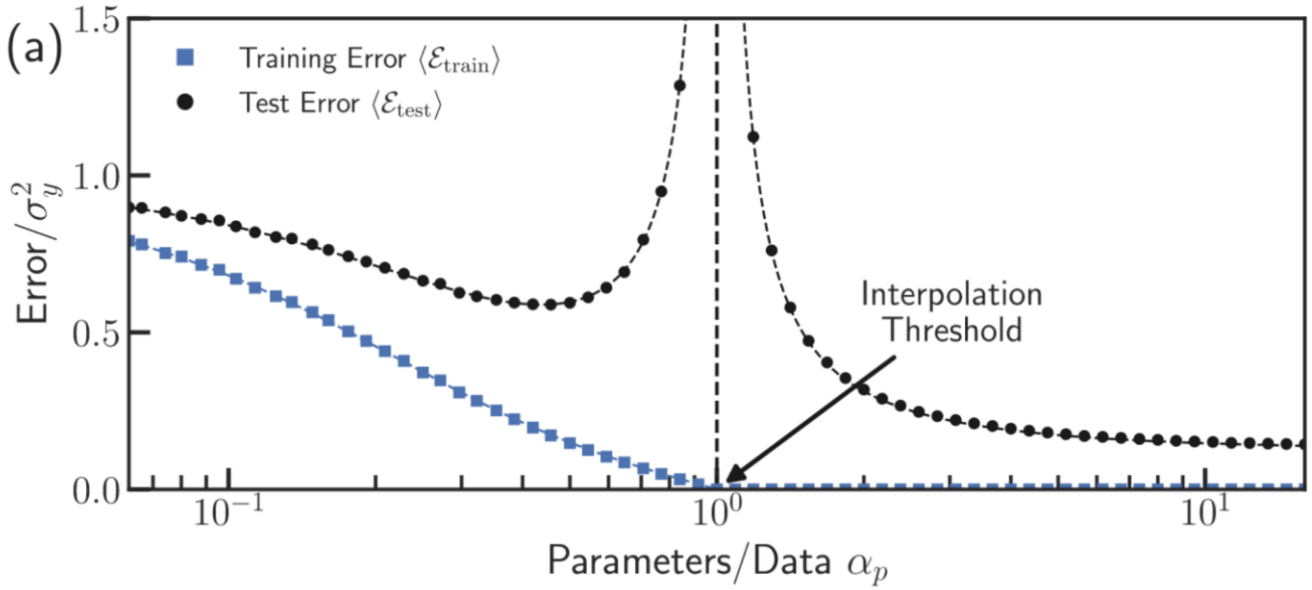
What is the role of **over-parametrization?**

In my opinion, machine learning is distinguished by the fact that

degrees of freedom(big models) $\gg$ # of big data.

We are able to interpolate data, yet still achieve good generalization.

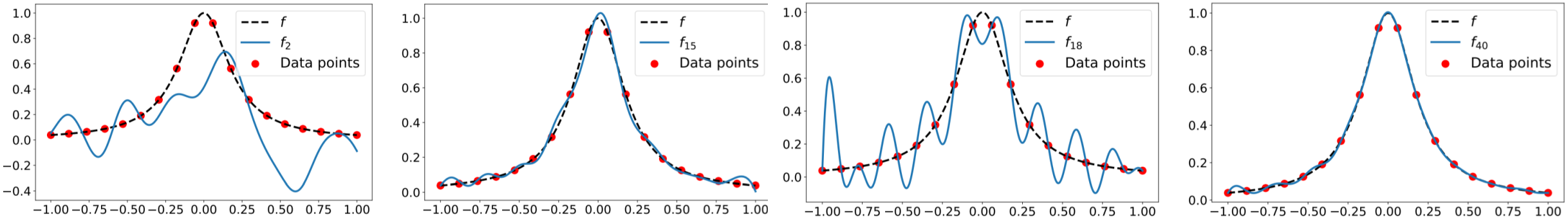
Does the **implicit bias underlying double descent** work in concert with derivative data?



Setting #1: **Surrogate Modeling** and Double Descent

When the data come from noiseless numerical simulations—e.g., data $y_i = f_{\text{PDE}}(x_i)$—why is double descent surprising?

Partial answer, from discussions with Jakob Zech: perhaps because classical we know *Lebesgue constant* of polynomial interpolants is high?



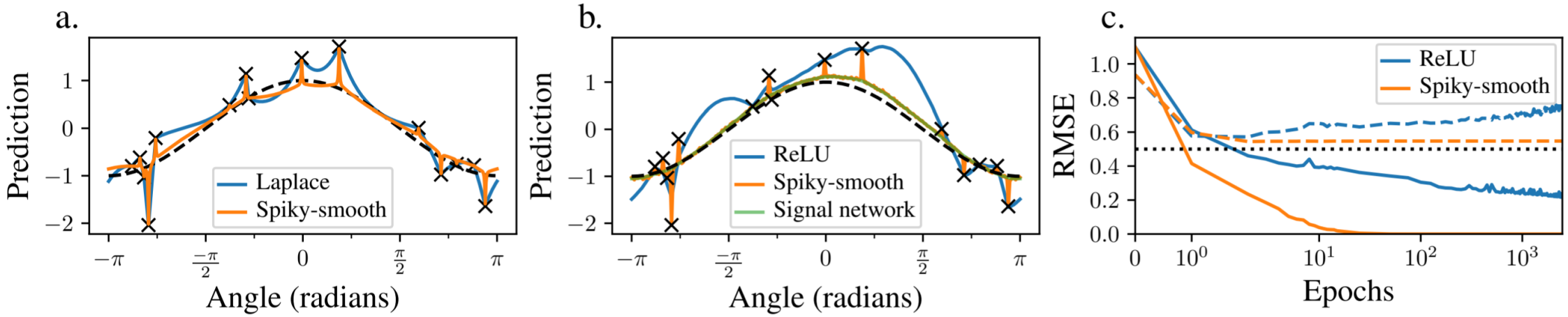
(a) $n = 2$ (underparameterization) (b) $n = 15$ (balance of appr. and gen. error) (c) $n = 18$ (interpolation threshold) (d) $n = 40$ (overparameterization)

Figure 15.4 from P. Petersen and J. Zech, "Mathematical theory of deep learning," Jul. 25, 2024 http://arxiv.org/abs/2407.18384

Setting #2: **Statistical Learning** and Double Descent

When data are noisy, double descent is surprising since it suggests models which interpolate noisy data—i.e., memorize the noise—achieve better generalization.

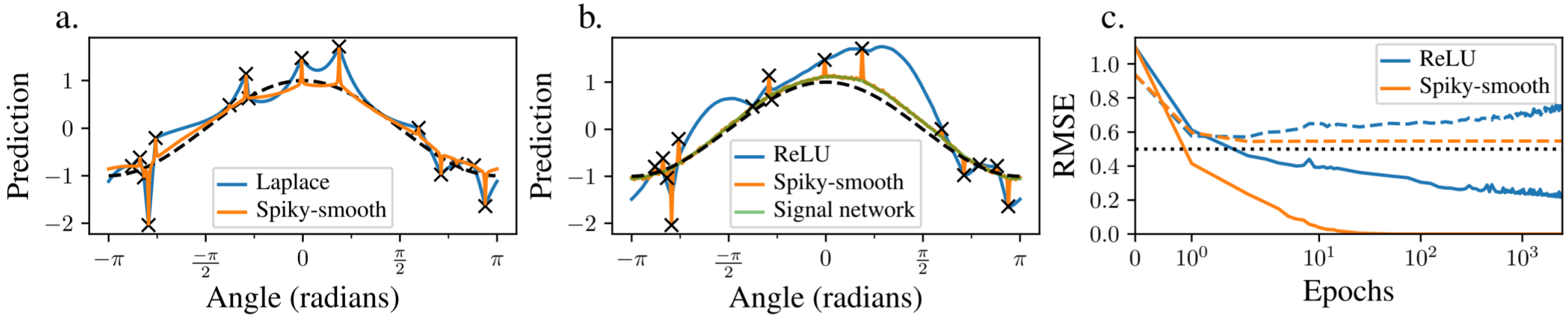
Intuitively, the models must look like "spike+smooth" interpolants.



M. Haas, D. Holzmüller, U. von Luxburg, and I. Steinwart, "Mind the spikes: Benign overfitting of kernels and neural networks in fixed dimension," Nov. 06, 2024, *arXiv*: arXiv:2305.14077. doi: 10.48550/arXiv.2305.14077.

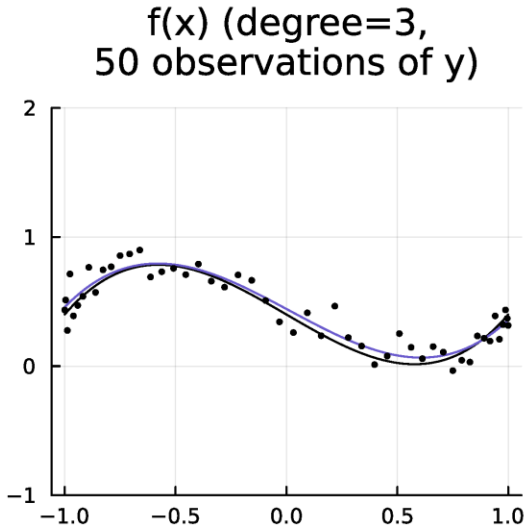
Setting #2: **Statistical Learning** and Double Descent

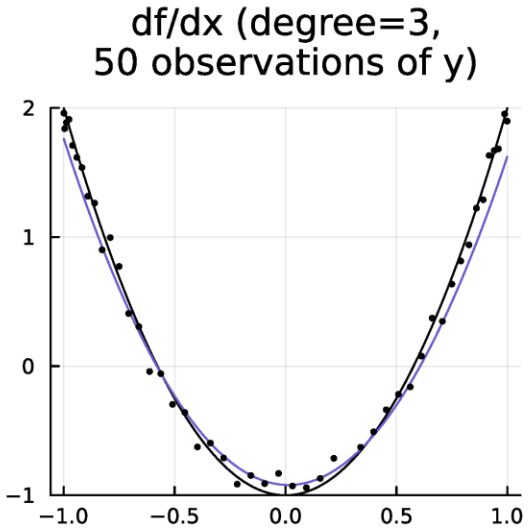
But if we train networks to **interpolate noisy function evaluations and also noisy derivatives**, what will they look like?

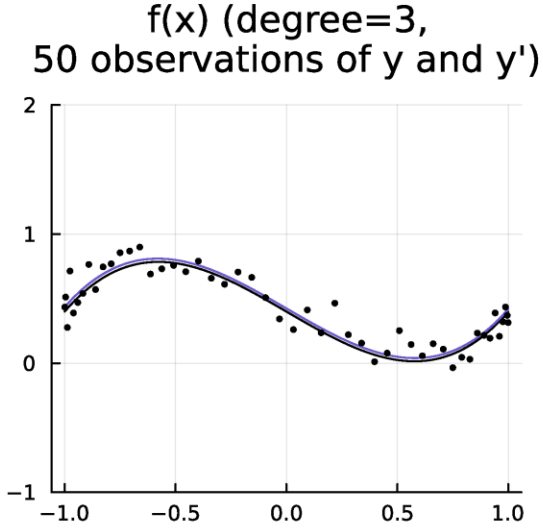


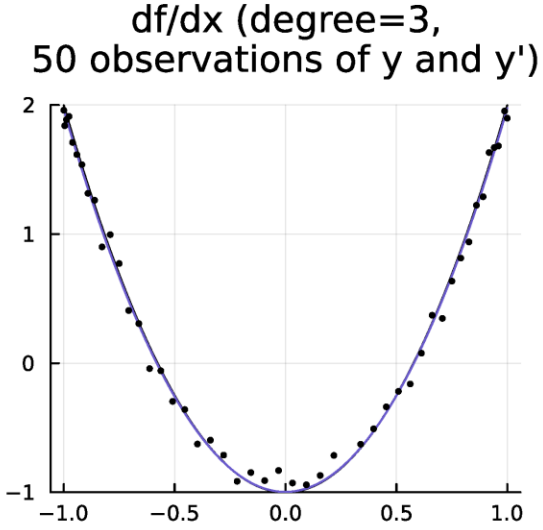
M. Haas, D. Holzmüller, U. von Luxburg, and I. Steinwart, "Mind the spikes: Benign overfitting of kernels and neural networks in fixed dimension," Nov. 06, 2024, *arXiv*: arXiv:2305.14077. doi: 10.48550/arXiv.2305.14077.

Interlude: Sobolev training with **polynomial features**









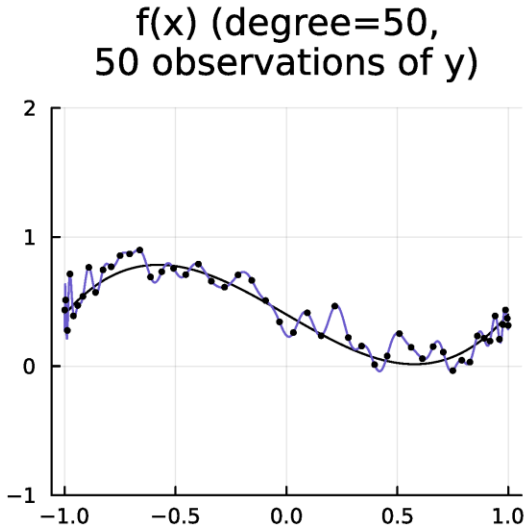
$$\text{Let } \begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$$

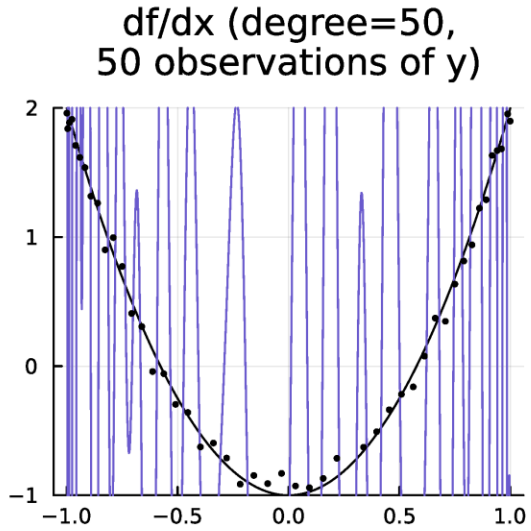
e.g., `import numpy.polynomial.legendre`, and find the **minimum norm interpolant**

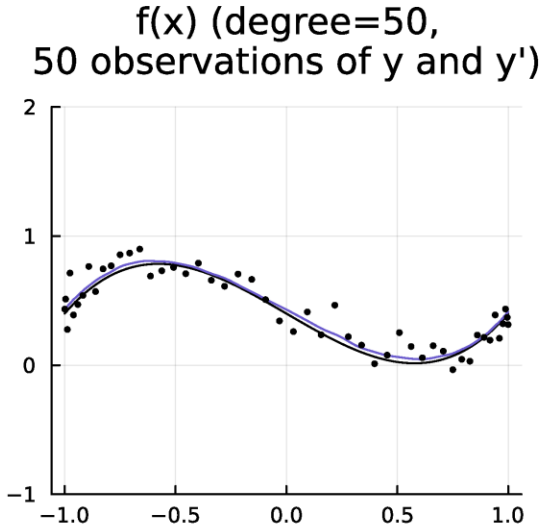
$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

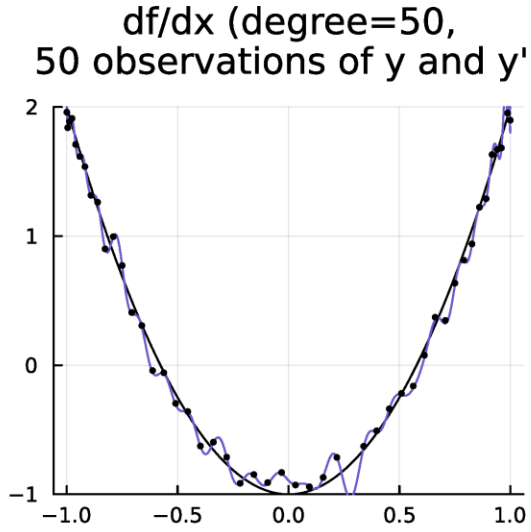
Note: figures are similar even with noise-less data.

Interlude: Sobolev training with **polynomial features**









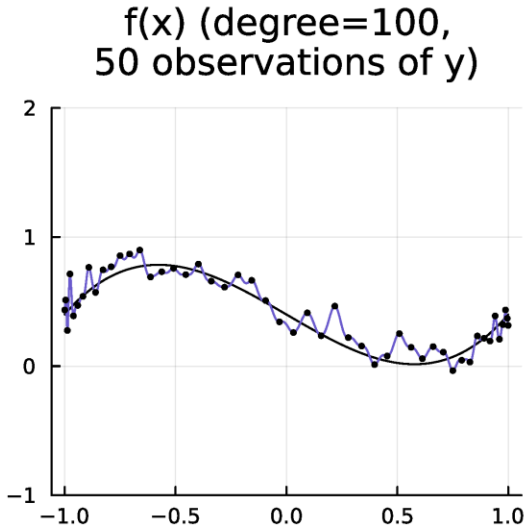
Let $\begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$

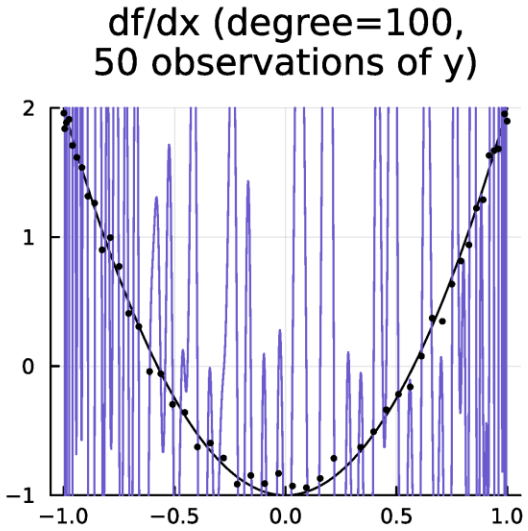
e.g., `import numpy.polynomial.legendre`, and find the **minimum norm interpolant**

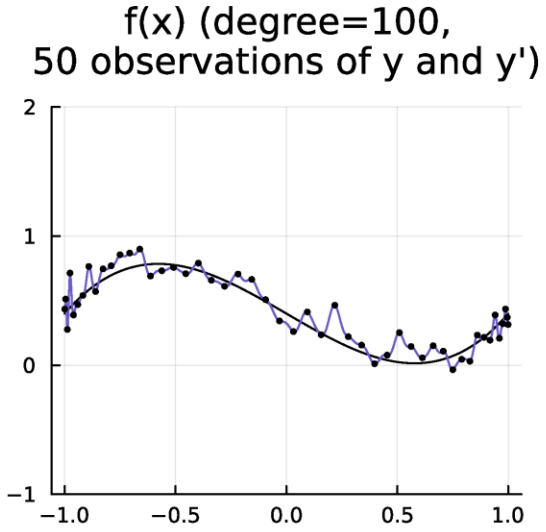
$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

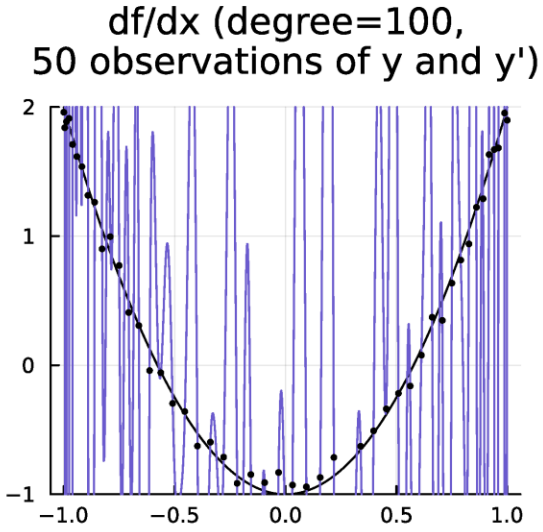
Note: figures are similar even with noise-less data.

Interlude: Sobolev training with **polynomial features**









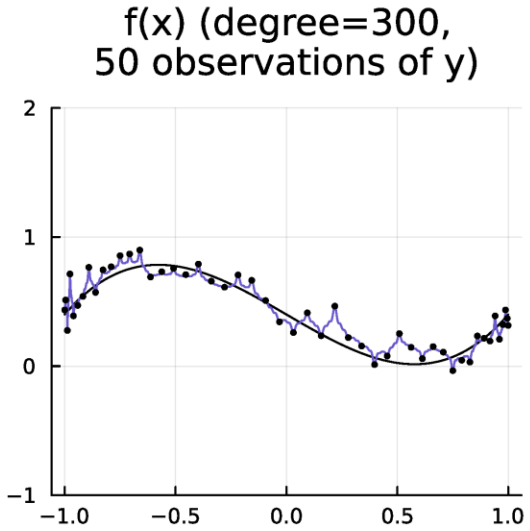
Let $\begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$

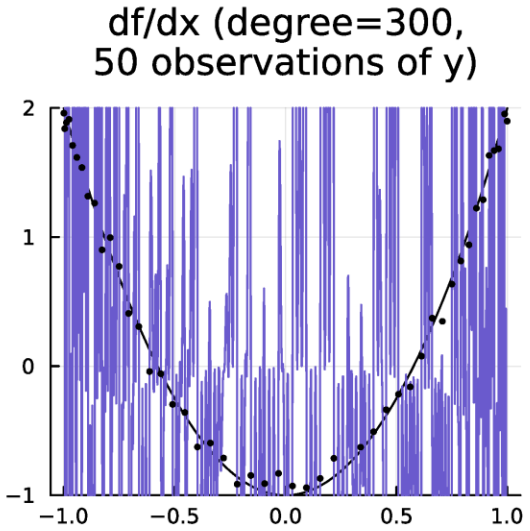
e.g., `import numpy.polynomial.legendre`, and find the **minimum norm interpolant**

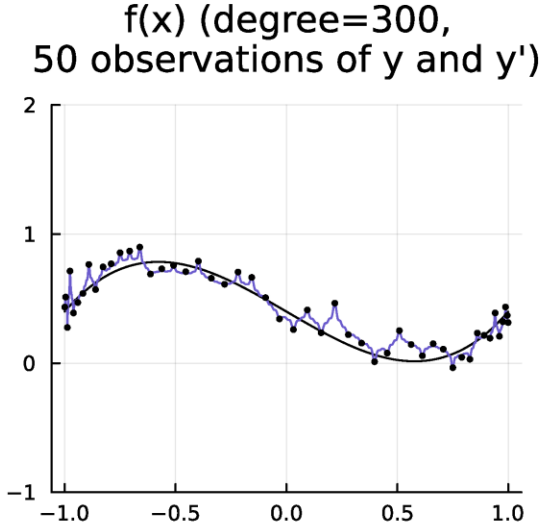
$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

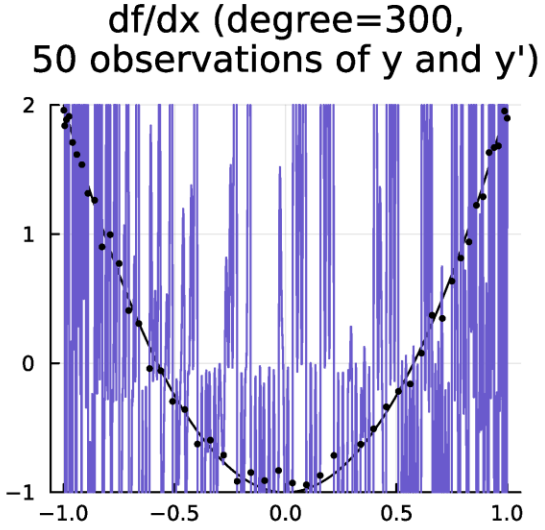
Note: figures are similar even with noise-less data.

Interlude: Sobolev training with **polynomial features**









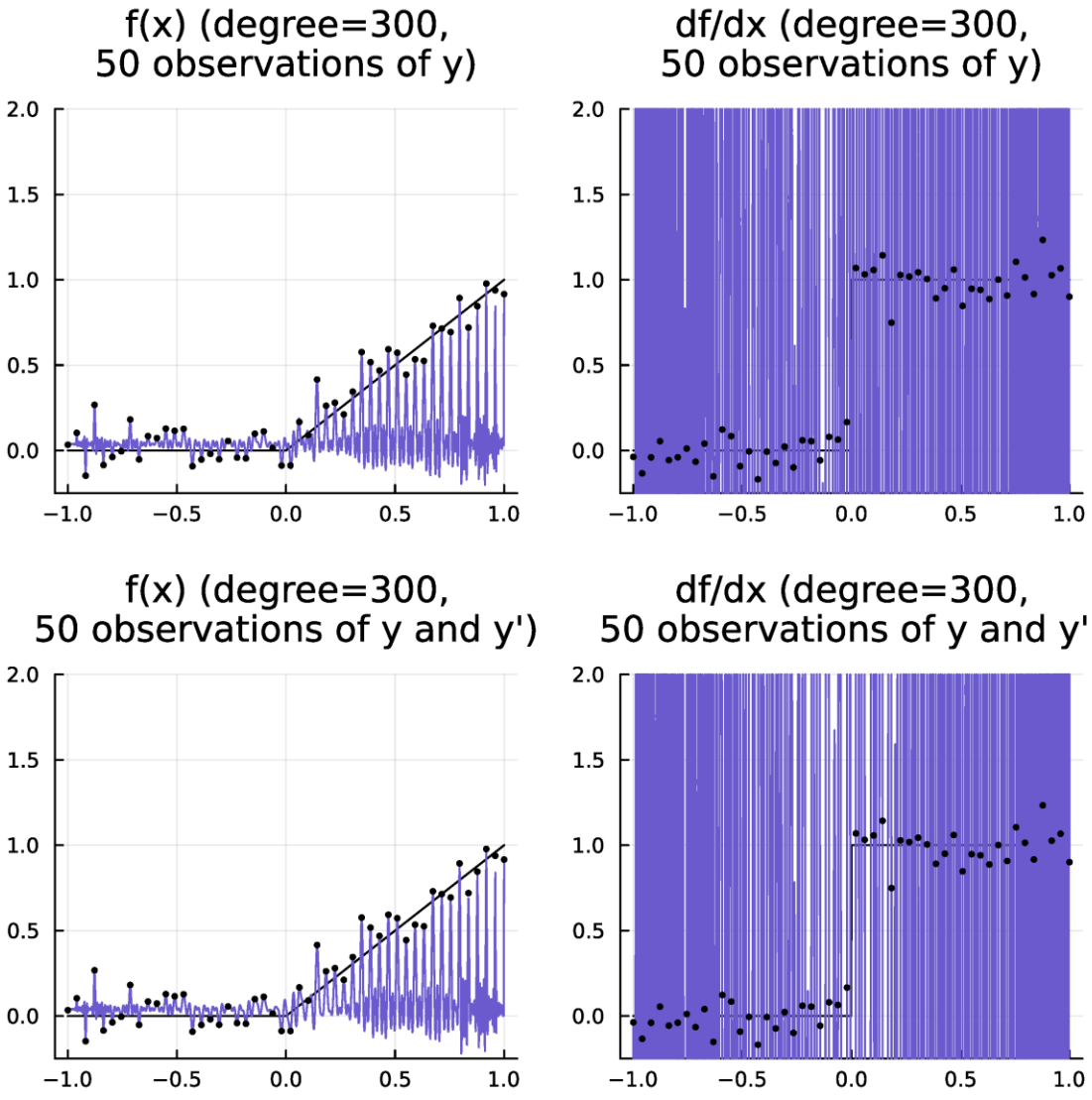
$$\text{Let } \begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$$

e.g., `import numpy.polynomial.legendre`, and find the **minimum norm interpolant**

$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

Note: figures are similar even with noise-less data.

Interlude: Sobolev training with **polynomial features**



Polynomials are difficult to study!

Recall if $\{\varphi_k(x)\}$ are orthogonal polynomials wrt μ, and if $f(x) = \sum_k w_k \varphi_k(x)$, then

$$\|w\|_2^2 = \sum_k w_k^2 = \int f(x)^2 \mathrm{d}\mu(x) \, .$$

Heuristically, with finite amount of data, the minimum norm interpolant will converge to the zero function with spikes as $p \to \infty$.

The Legendre polynomials in previous slides were not normalized!

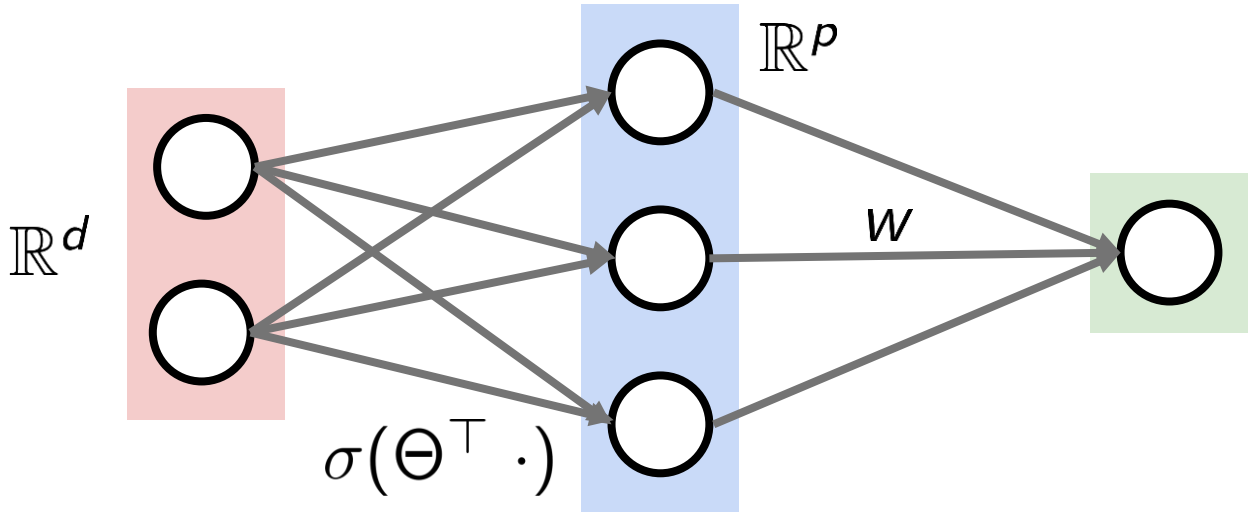
Single hidden-layer **random feature models**

A non-trivial parametric class is single hidden-layer neural networks

$$f_w(x) = \langle w, \sigma(\tfrac{1}{\sqrt{d}}\Theta^\top x)\rangle,$$

with input $x \sim \mathcal{N}(0, I_d)$ and random features $\Theta \in \mathbb{R}^{d\times p}$, where $\Theta_{:,i} \sim \mathcal{N}(0, I_d)$.

We take the **proportional asymptotics limit** $d, n, p \to \infty$ with $\alpha = n/p$ and $\gamma = d/p$ fixed. (These limits do not commute!).



Architecture
d = input dim
n = training pts
p = num features

A massive simplification of **neural networks** in practice

The pre-activation features Θ are **fixed** during training. Random feature [RR08] and neural tangent kernels [JGH18] fall in the "lazy-learning regime" since

$$\frac{\partial}{\partial w_k} f_w(x) = \sigma\left(\frac{1}{\sqrt{d}}\theta^\top x\right) \qquad \sim O(1),$$

$$\nabla_\theta f_w(x) = \frac{1}{\sqrt{d}} x\sigma'(\theta^\top x) \qquad \sim O(d^{-1/2}),$$

A massive simplification of **neural networks** in practice

The pre-activation features Θ are **fixed** during training. Random feature [RR08] and neural tangent kernels [JGH18] fall in the "lazy-learning regime" since

$$\frac{\partial}{\partial w_k} f_w(x) = \sigma\left(\frac{1}{\sqrt{d}}\theta^\top x\right) \qquad \sim O(1),$$
$$\nabla_\theta f_w(x) = \frac{1}{\sqrt{d}} x \sigma'(\theta^\top x) \qquad \sim O(d^{-1/2}),$$

so gradient descent with step size $\eta \sim O(1/\sqrt{d})$

$$w_k^{(t+1)} = w_k^{(t)} - \eta\, \partial_{w_k} f_w(x)$$
$$\theta^{(t+1)} = \theta^{(t)} - \eta\, \partial_\theta f_w(x)$$

does not meaningfully change $|\Theta|_F$ (although $\|\Theta\|_{\text{op}}$ can change! [BES+22]).

Non-rigorous theoretical predictions

The assumption of proportional asymptotics—big models, big data, and big inputs—is what makes theory possible [MM22, GLR+20, HL20].

With linear asymptotics $\alpha = n/p$ and $\gamma = d/p$, everything becomes essentially Gaussian: this builds on the work of El Karoui, and Pennington and Worah.

- N. E. Karoui, "The spectrum of kernel random matrices,"
- J. Pennington and P. Worah, "Nonlinear random matrix theory for deep learning"
- S. Mei and A. Montanari, "The Generalization Error of Random Features Regression: Precise Asymptotics and the Double Descent Curve"
- S. Goldt, B. Loureiro, G. Reeves, F. Krzakala, M. Mézard, and L. Zdeborová, "The Gaussian equivalence of generative models for learning with shallow neural networks".
- H. Hu and Y. M. Lu, "Universality Laws for High-Dimensional Learning with Random Features."

Non-rigorous theoretical predictions

The assumption of proportional asymptotics—big models, big data, and big inputs—is what makes theory possible [MM22, GLR+20, HL20].

With linear asymptotics $\alpha = n/p$ and $\gamma = d/p$, everything becomes essentially Gaussian: this builds on the work of El Karoui, and Pennington and Worah.

Further assuming the manifold hypothesis—here, data drawn a single-index teacher $y = \varphi(\langle \theta_0, x \rangle)$ with $\theta_0 \sim \mathcal{N}(0, \frac{1}{d} I)$—then **replica method from statistical physics** yields non-rigorous theoretical error predictions.

- N. E. Karoui, "The spectrum of kernel random matrices,"
- J. Pennington and P. Worah, "Nonlinear random matrix theory for deep learning"
- S. Mei and A. Montanari, "The Generalization Error of Random Features Regression: Precise Asymptotics and the Double Descent Curve"
- S. Goldt, B. Loureiro, G. Reeves, F. Krzakala, M. Mézard, and L. Zdeborová, "The Gaussian equivalence of generative models for learning with shallow neural networks".
- H. Hu and Y. M. Lu, "Universality Laws for High-Dimensional Learning with Random Features."

Non-rigorous theoretical predictions

The assumption of proportional asymptotics—big models, big data, and big inputs—is what makes theory possible [MM22, GLR+20, HL20].

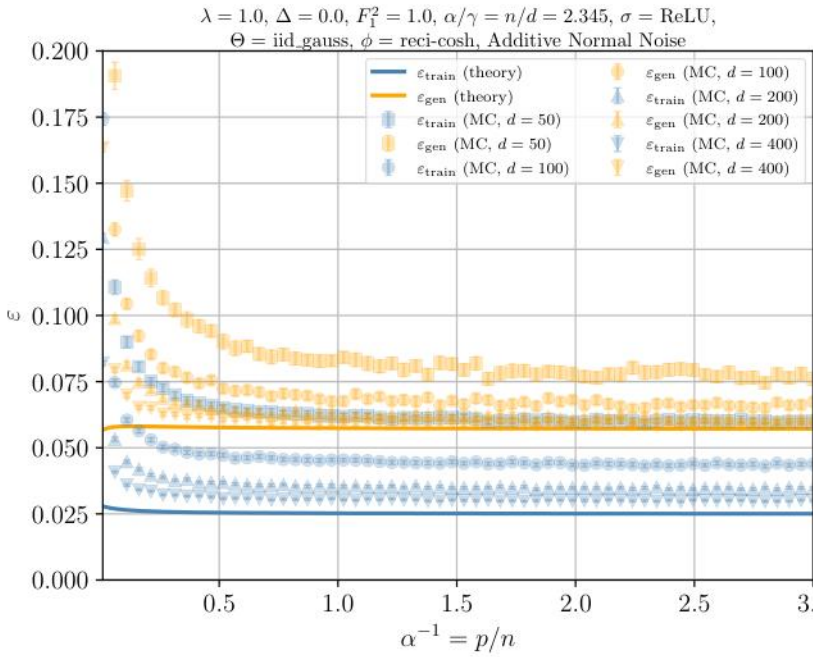
With linear asymptotics $\alpha = n/p$ and $\gamma = d/p$, everything becomes essentially Gaussian: this builds on the work of El Karoui, and Pennington and Worah.

Further assuming the manifold hypothesis—here, data drawn a single-index teacher $y = \varphi(\langle \theta_0, x \rangle)$ with $\theta_0 \sim \mathcal{N}(0, \frac{1}{d} I)$—then **replica method from statistical physics** yields non-rigorous theoretical error predictions.

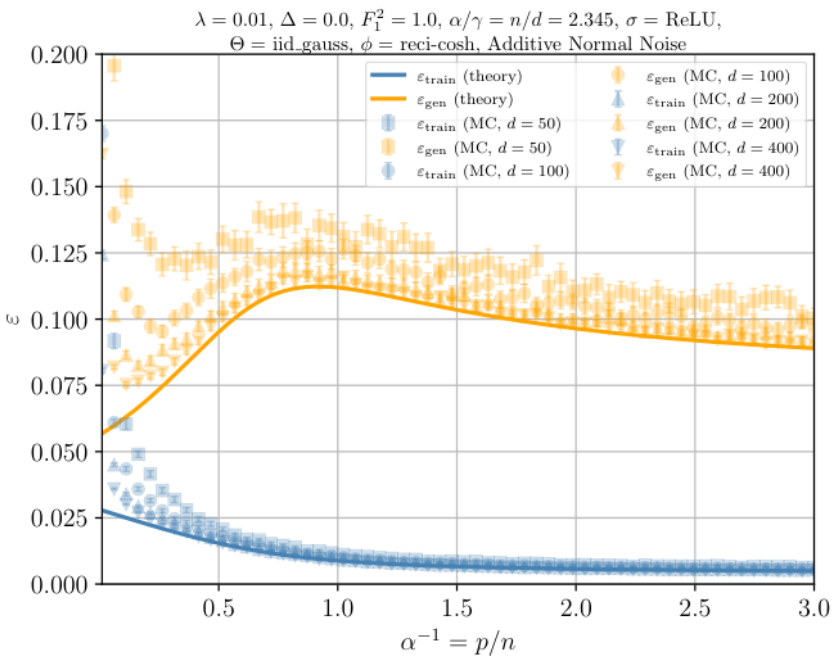
To obtain a fully asymptotic description, the calculations require relatively modern tools from RMT, e.g., operator valued free probability and Stiejes transforms [RS]. **These asymptotics kick in remarkably quickly.**

- N. E. Karoui, "The spectrum of kernel random matrices,"
- J. Pennington and P. Worah, "Nonlinear random matrix theory for deep learning"
- S. Mei and A. Montanari, "The Generalization Error of Random Features Regression: Precise Asymptotics and the Double Descent Curve"
- S. Goldt, B. Loureiro, G. Reeves, F. Krzakala, M. Mézard, and L. Zdeborová, "The Gaussian equivalence of generative models for learning with shallow neural networks".
- H. Hu and Y. M. Lu, "Universality Laws for High-Dimensional Learning with Random Features."

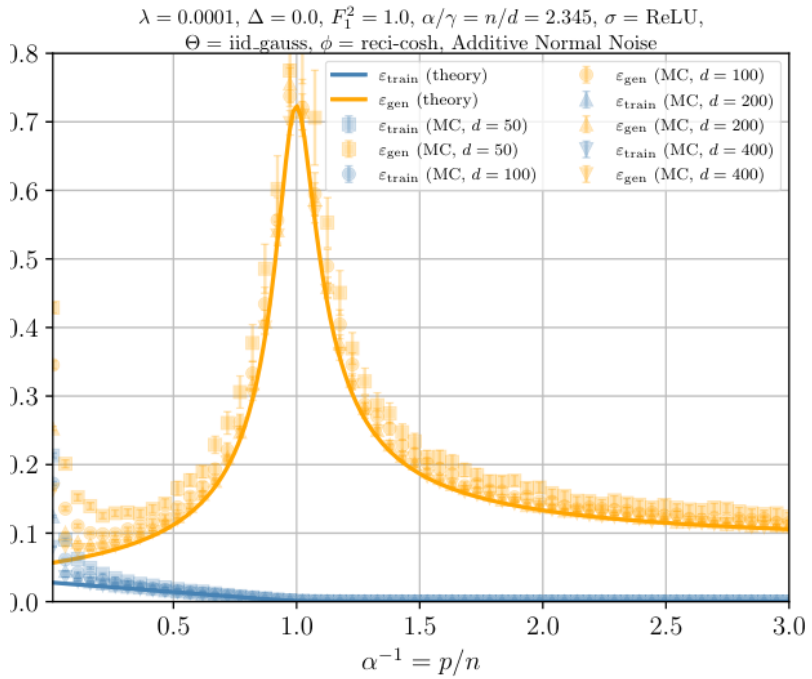
Theoretical predictions for **standard L2 training**



(a) Regularization $\lambda = 10^0$, iid Gaussian features



(c) Regularization $\lambda = 10^{-2}$, iid Gaussian features

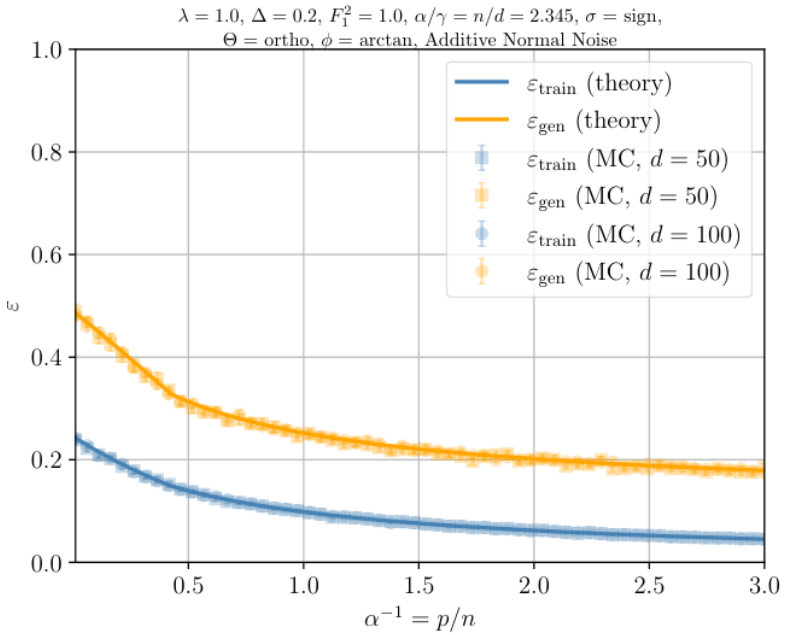


(e) Regularization $\lambda = 10^{-4}$, iid Gaussian features

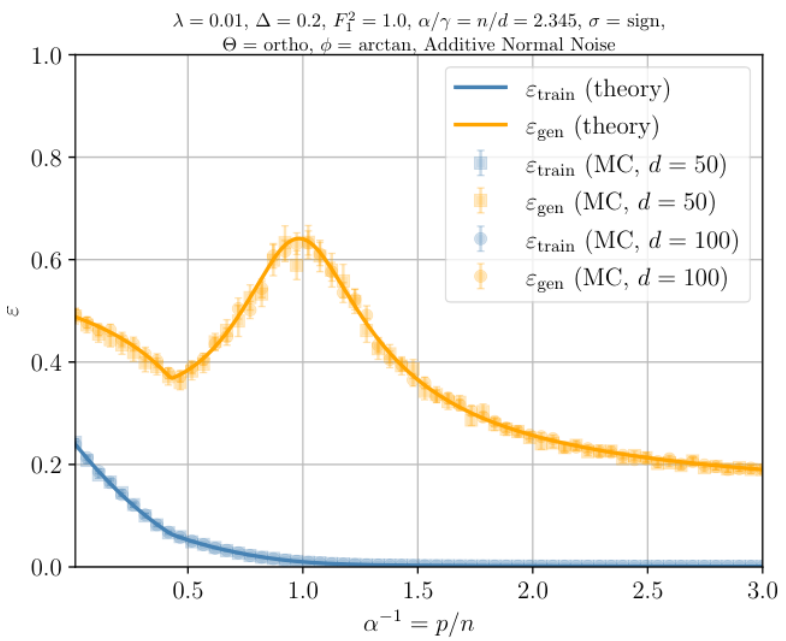
$$\varepsilon_{\text{train}} = \lim_{d,n,p\to\infty} \min_{w\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}(y_i - \langle w, \sigma(\Theta^\top x_i)\rangle)^2 + \lambda\|w\|^2$$

$$\varepsilon_{\text{tst}}^{L_2} = \mathbb{E}_{X,Y\sim p_{\text{data}}}[(Y - f_{w^{\text{opt}}}(X))^2]$$

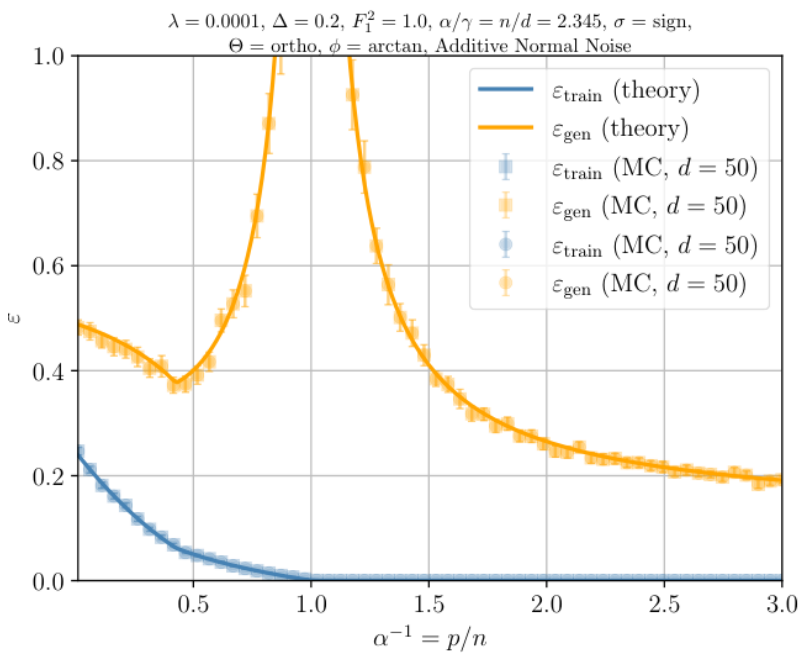
Theoretical predictions for **standard L2 training**



(a) Regularization $\lambda = 10^0$, activation $\sigma = \text{sgn}$



(c) Regularization $\lambda = 10^{-2}$, activation $\sigma = \text{sgn}$



(e) Regularization $\lambda = 10^{-4}$, activation $\sigma = \text{sgn}$

Mei and Montanari showed that there are instances where it is preferable to over-parametrized and over-fit the noise to get better generalization.

Sobolev training with random feature models

In order to apply the replica method to study Sobolev training, we must must **project gradient data onto low-dimensional subspaces**.

This was already described in Czarnecki et al., and commonly done in contemporary Sobolev training, see e.g. derivative informed neural operators (DINO).

A potential concern is that this optimisation might be expensive when either the output dimensionality of f or the order K are high, however one can reduce this cost through stochastic approximations. Specifically, if f is a multivariate function, instead of a vector gradient, one ends up with a full Jacobian matrix which can be large. To avoid adding computational complexity to the training process, one can use an efficient, stochastic version of Sobolev Training: instead of computing a full Jacobian/Hessian, one just computes its projection onto a random vector (a direct application of a known estimation trick [19]). In practice, this means that during training we have a random variable v sampled uniformly from the unit sphere, and we match these random projections instead:

$$\sum_{i=1}^{N}\left[\ell(m(x_i|\theta),f(x_i))+\sum_{j=1}^{K}\mathbb{E}_{v^j}\left[\ell_j\left(\left\langle D^j_{\mathbf{x}}m(x_i|\theta),v^j\right\rangle,\left\langle D^j_{\mathbf{x}}f(x_i),v^j\right\rangle\right)\right]\right]. \tag{2}$$

Sobolev training with random feature models

Taking $V_k \in \mathbb{R}^d$ with unit columns $V_k \sim \mathcal{N}(0, \frac{1}{d} I_d)$ and minimizing

$$\varepsilon_{\text{train}}^{H_1^k} = \lim_{d,n,p\to\infty} \min_{w\in\mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left\{ (y_i - f_w(x_i))^2 + \sum_k \| V_k^\top (y_i' - \nabla_x f_w(x_i)) \|^2 \right\} + \lambda \|w\|^2$$

results in a **mismatch in scales**. Note $k \in \mathbb{N}$ is fixed.

Sobolev training with random feature models

Taking $V_k \in \mathbb{R}^d$ with unit columns $V_k \sim \mathcal{N}(0, \frac{1}{d} I_d)$ and minimizing

$$\varepsilon_{\text{train}}^{H_1^k} = \lim_{d,n,p\to\infty} \min_{w\in\mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left\{ (y_i - f_w(x_i))^2 + \sum_k \| V_k^\top (y_i' - \nabla_x f_w(x_i)) \|^2 \right\} + \lambda \|w\|^2$$

results in a **mismatch in scales**. Note $k \in \mathbb{N}$ is fixed.

If $\theta_0 \sim \mathcal{N}(0, \frac{1}{d} I_d)$ and $x \sim \mathcal{N}(0, I_d)$, then

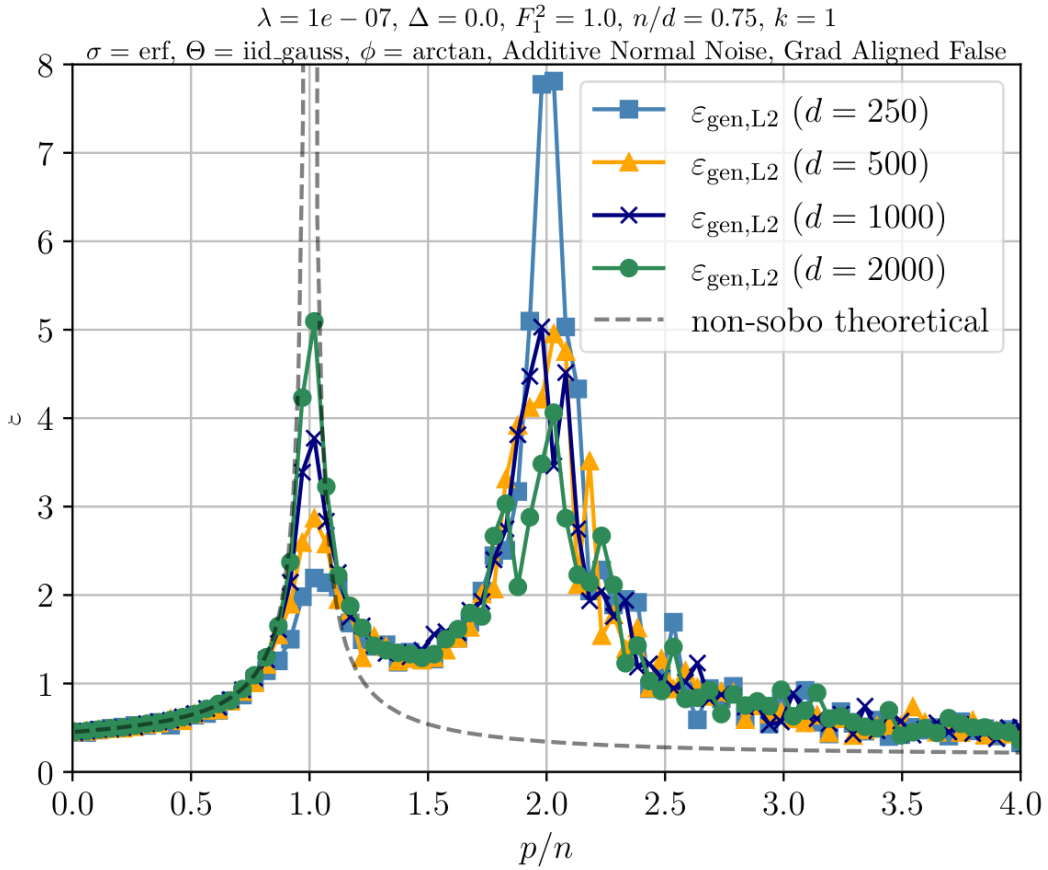
$$y = \varphi(\langle \theta_0, x \rangle) = O_d(1) \,,$$

whereas

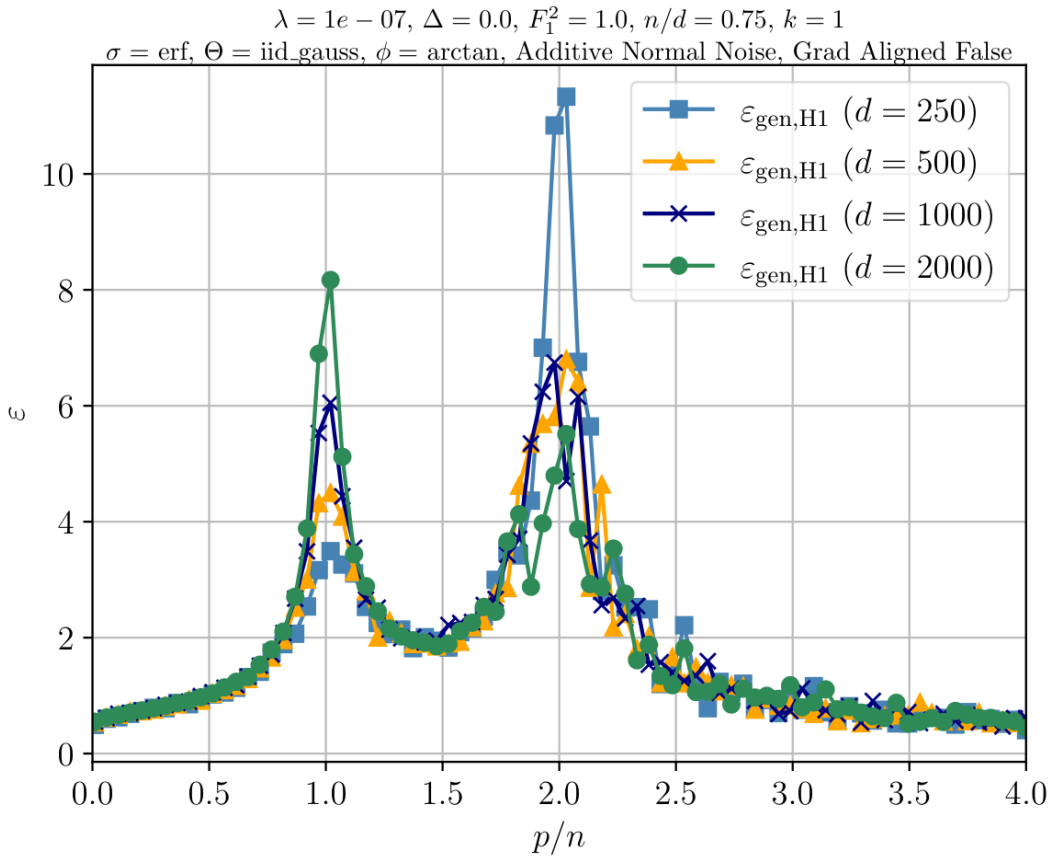
$$V_k^\top y_i' = V_k^\top \theta_0 \cdot \varphi'(\langle \theta_0, x \rangle) \sim O_d(d^{-1/2}),$$

meaning asymptotically you "never see the gradients."

Trivial universality with wrong gradient scaling



(a) $k = 1$, L_2 generalization



(b) $k = 1$, H_1 generalization

Gradient projection model

Instead, we must consider $V_k \sim \mathcal{N}(0, I)$ so that $|V_k| = O(\sqrt{d})$ and

$$\varepsilon_{\text{train}}^{H_1^k} = \lim_{d,n,p\to\infty} \min_{w\in\mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left\{ (y_i - f_w(x_i))^2 + \sum_k \| V_k^\top (y_i' - \nabla_x f_w(x_i)) \|^2 \right\} + \lambda \|w\|^2 \ .$$

(Aside: can also consider *informed* gradient projection.)

Gradient projection model

Instead, we must consider $V_k \sim \mathcal{N}(0, I)$ so that $|V_k| = O(\sqrt{d})$ and

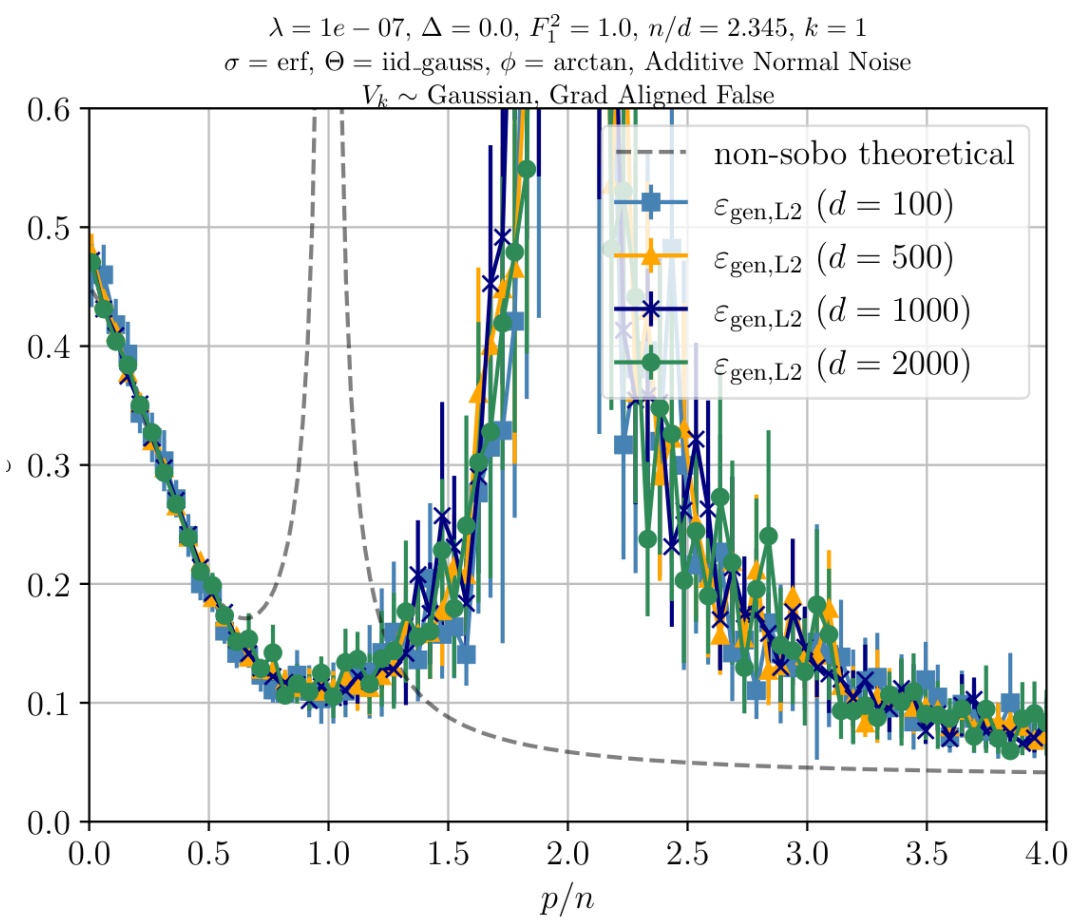
$$\varepsilon_{\text{train}}^{H_1^k} = \lim_{d,n,p\to\infty} \min_{w\in\mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left\{ (y_i - f_w(x_i))^2 + \sum_k \| V_k^\top (y_i' - \nabla_x f_w(x_i)) \|^2 \right\} + \lambda \|w\|^2 \ .$$

(Aside: can also consider *informed* gradient projection.)

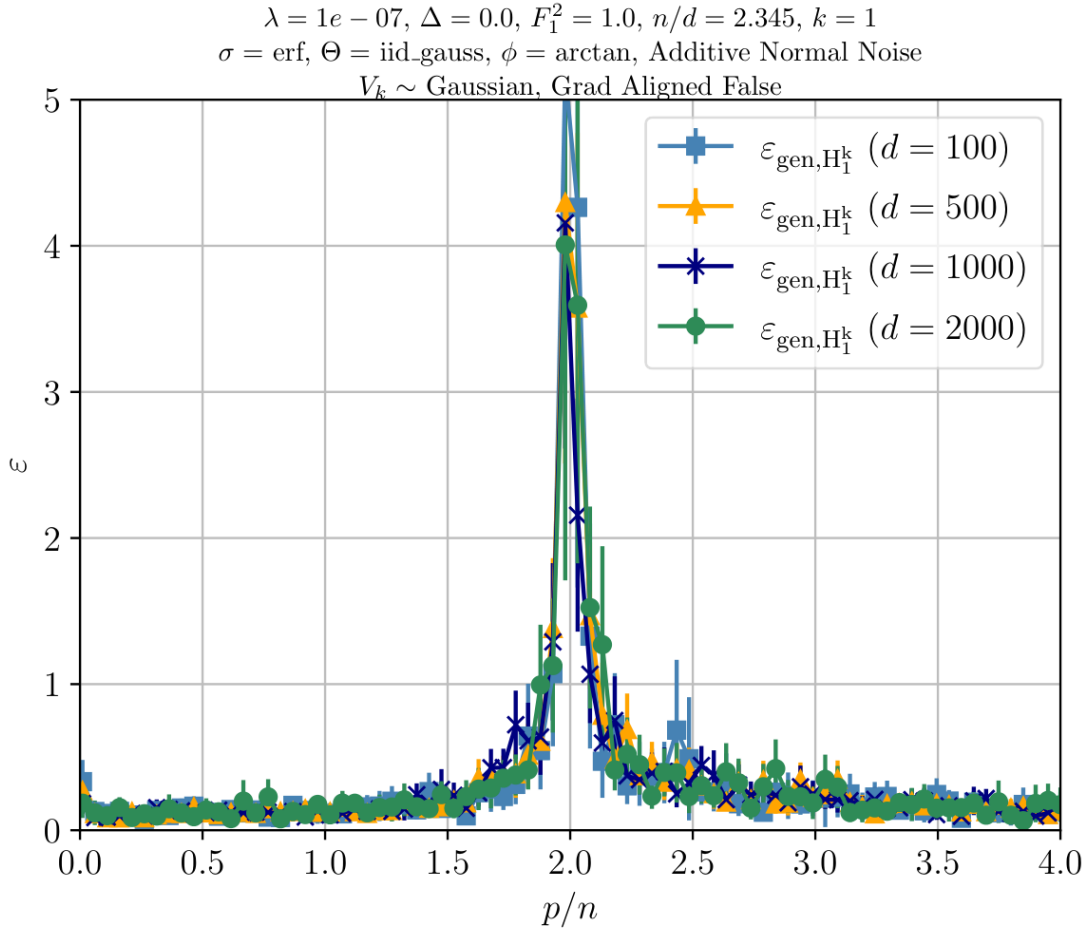
However, unlike L_2 training, the generalization and training error **do not converge**, but are random variables depending the alignment between the teacher vector and the gradient projection

$$\varpi_k := V_k^\top \theta_0 \sim \mathcal{N}(0, 1).$$

k=1 dimensional subspace projections



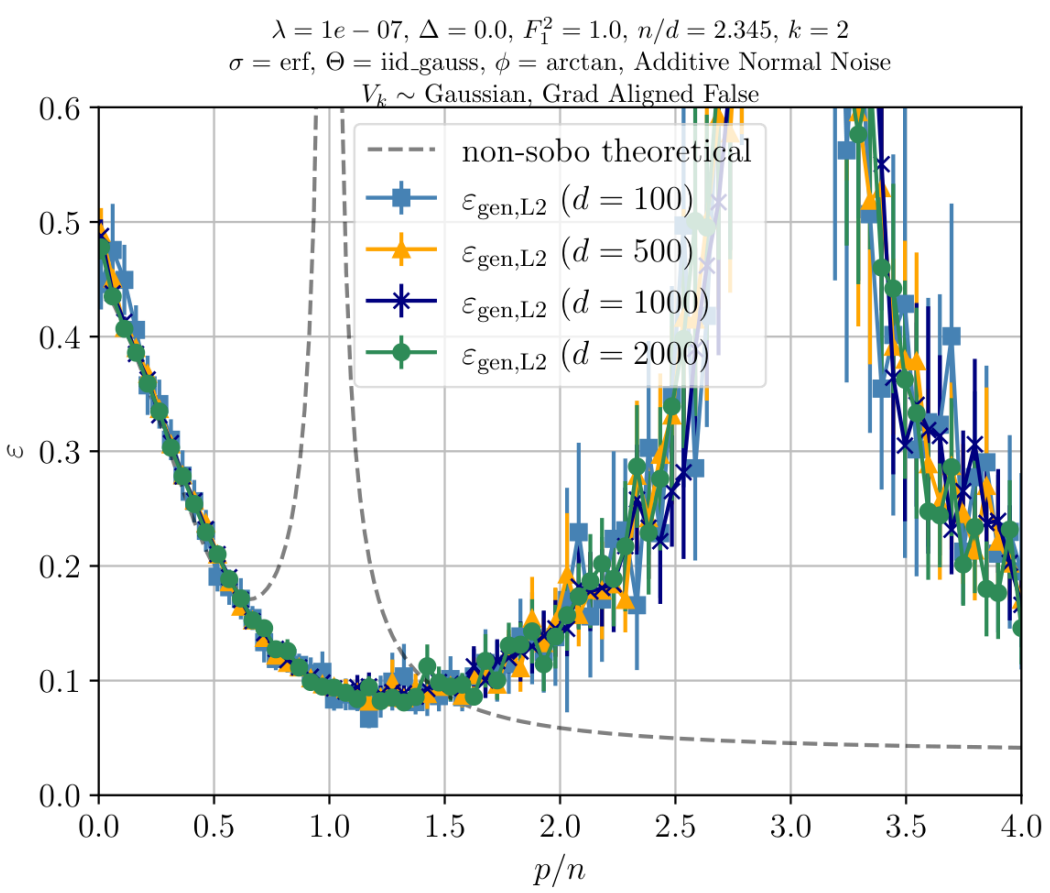
(a) $k = 1$, L_2 generalization



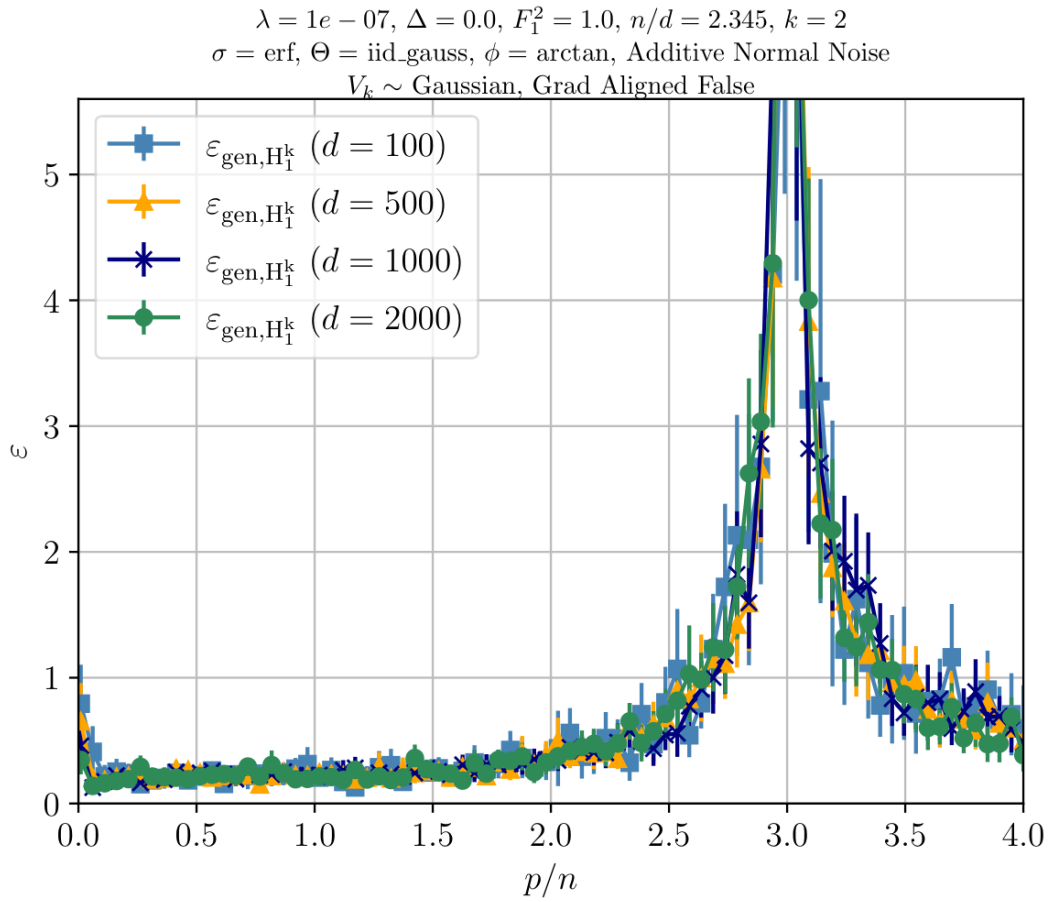
(b) $k = 1$, H_1 generalization

The phase transition shifts when you add derivative data!

k=2 dimensional subspace projections



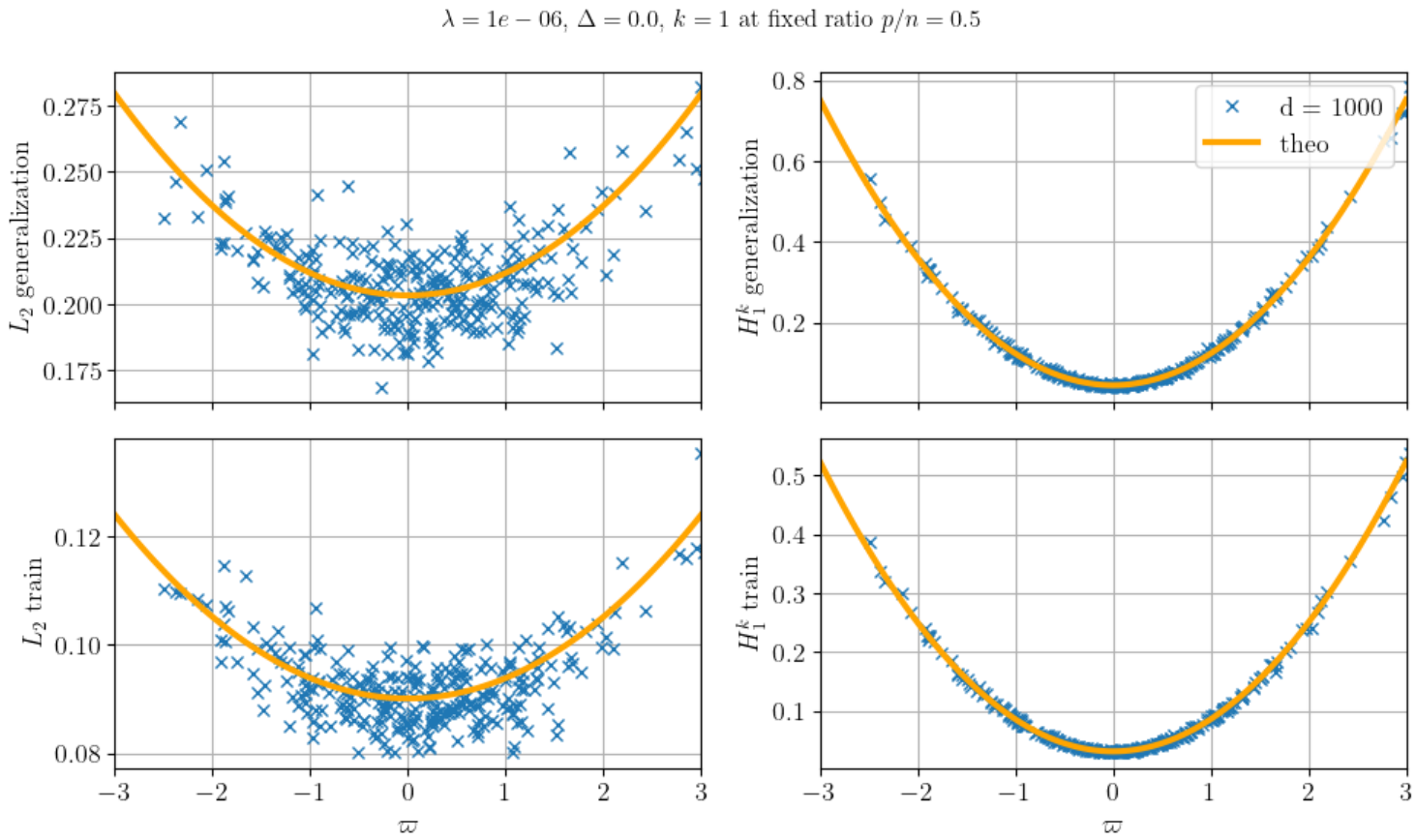
(c) $k = 2$, L_2 generalization



(d) $k = 2$, H_1 generalization

The phase transition shifts to p/n=k+1 when you add more derivative data...

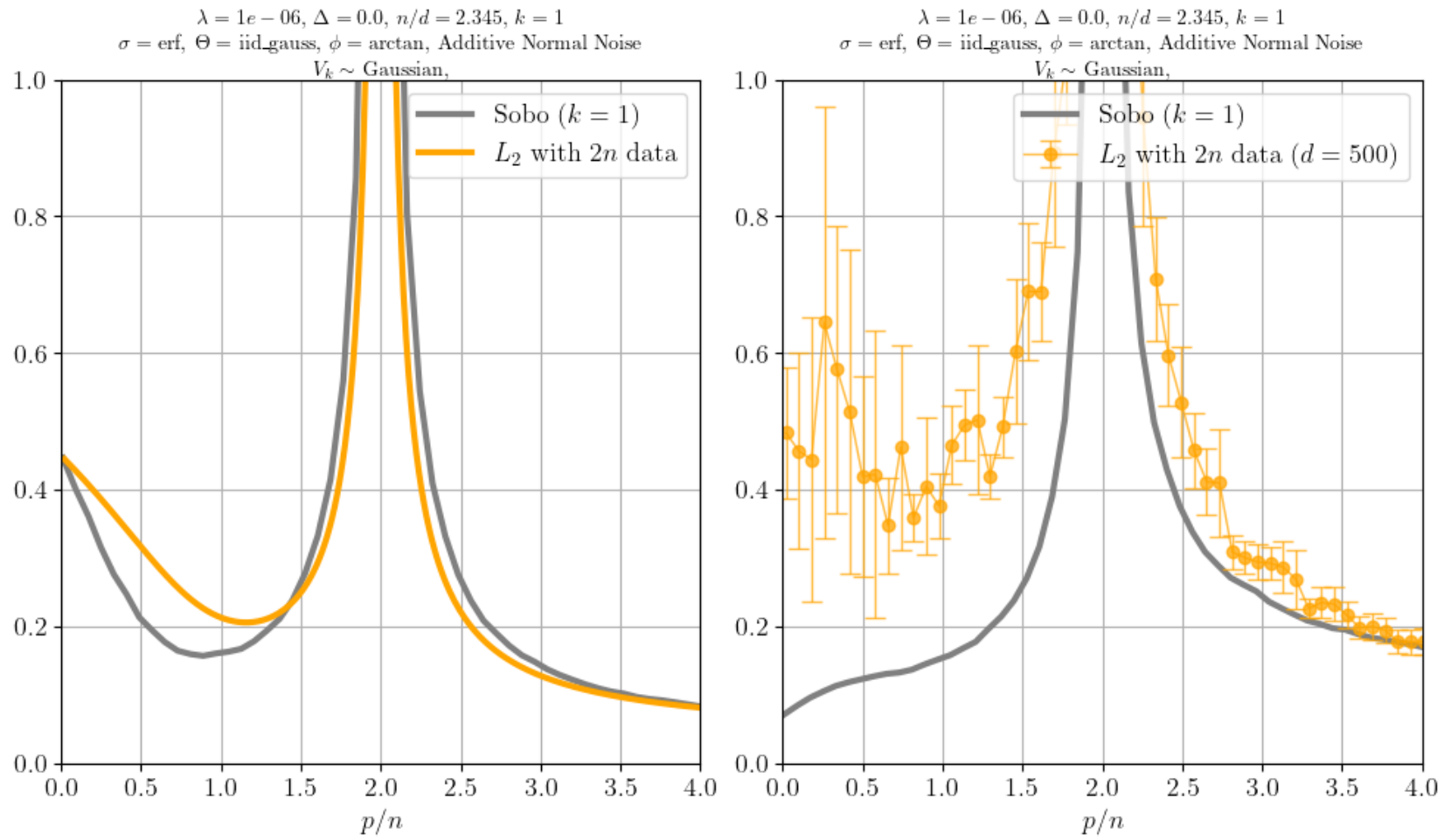
Error distribution with respect to alignment ϖ



The errors behave **quadratically** with respect to ϖ, i.e., the distribution is χ^2-distributed $\implies$ more alignment is worse?

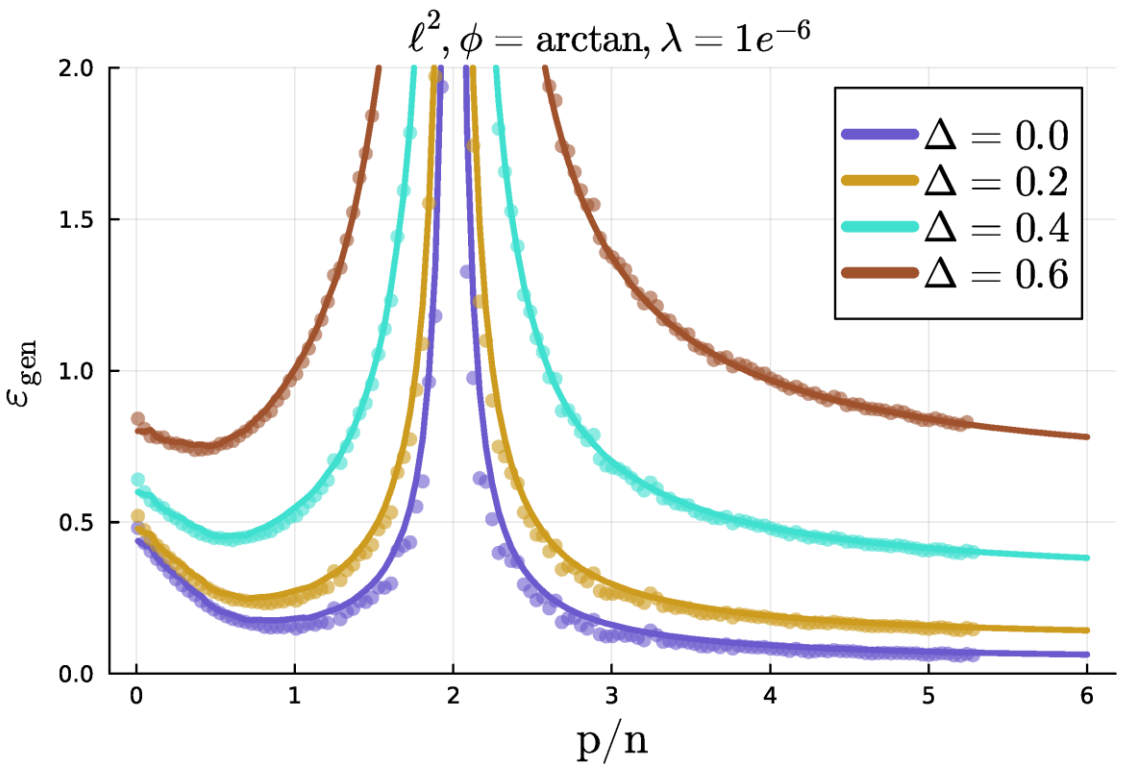
Although this is non-standard, we can compute this with the replica method.

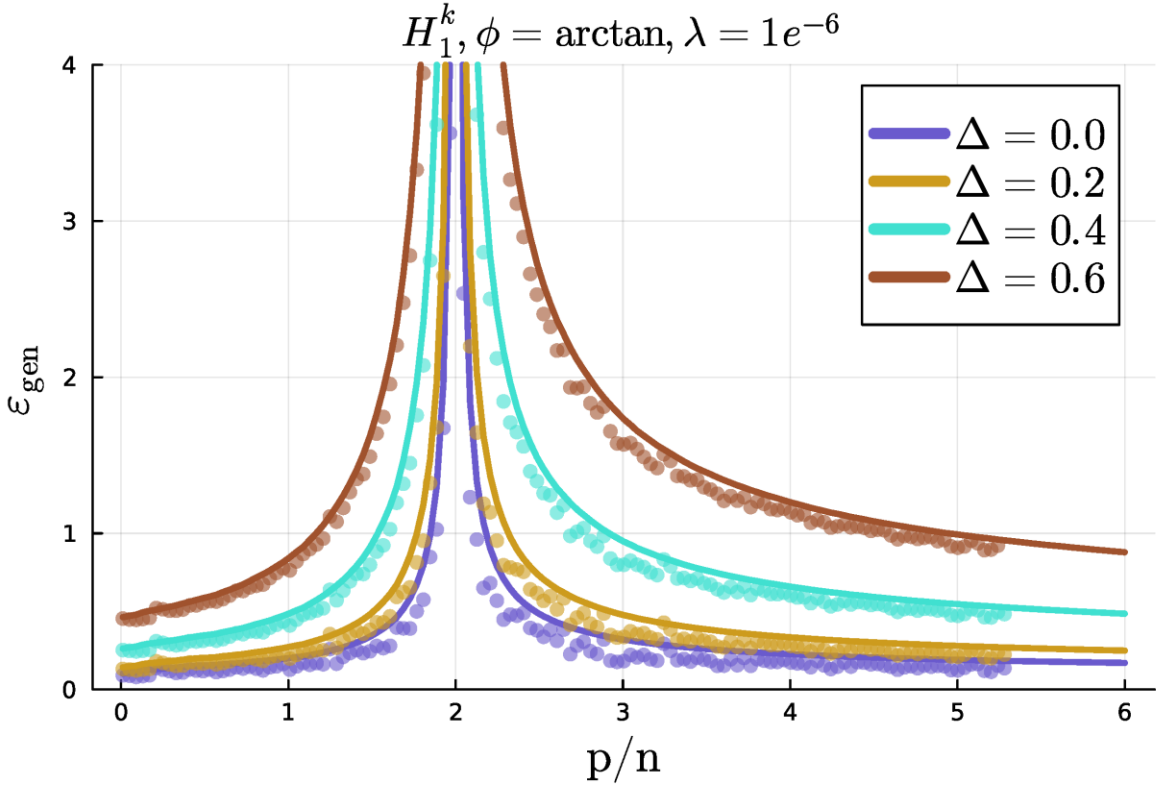
Normalizing by computational cost



Perhaps a fairer comparison is the **normalize by model cost**.

Effect of **noise** on function and gradient data





In spite of noisy data, it can be beneficial to **memorize the noise**.

Our analysis also shows that correlation in noise doesn't matter.

The main **technical tools**

The key technical idea is the **Gaussian Equivalence Theorem**, building on the seminal works of El Karoui (2010) and Pennington and Worah (2017).

Morally speaking, in the proportional asymptotics limit, single hidden-layer networks "look jointly Gaussian" with

$$\begin{pmatrix} \langle \theta_0, x \rangle \\ w^* \sigma(\Theta^\top x) \end{pmatrix} \overset{d}{=} \begin{pmatrix} \langle \theta_0, x \rangle \\ \kappa_0 \langle w^*, 1_p \rangle + \kappa_1 \langle w^*, \Theta^\top x \rangle + \kappa_* \eta \end{pmatrix},$$

where $\eta \sim \mathcal{N}(0,1)$, $\kappa_0 = \mathbb{E}_\eta[\sigma(\eta)]$, $\kappa_1 = \mathbb{E}_\eta[\sigma'(\eta)]$, and $\kappa_* = \mathbb{E}_\eta[\sigma^2(\eta)] - \kappa_0^2 - \kappa_1^2$.

Formal Sketch of Calculations

Defining $\omega = \langle \theta_0, x \rangle$, we can then write

$$\varepsilon_{\text{gen}} = \mathbb{E}_{x,y\sim p_{\text{data}}}[(y - f_{w^*}(x))^2] = \mathbb{E}_{\omega,\hat{y}\sim\mathcal{N}((0,\kappa_0 b)^\top,\Sigma)}[(\varphi(\omega) - \hat{y})^2],$$

evaluated at the optimal readout weights

$$w^* = (\kappa_1^2 X^\top \Theta^\top \Theta X + \kappa_*^2 I)^{-1} y \ .$$

The generalization error has a closed form asymptotic formulae related to the **Stieljes/Cauchy transform** describing the limiting eigenvalue distribution of the random matrix $\Theta^\top \Theta$ [DW18,MM20].

Replica setup for Sobolev training

Define the Hermite coefficients of the activation function σ and its derivative σ'

$$\kappa_0 = \mathbb{E}\left[\sigma(\xi)\right], \quad \kappa_1 = \mathbb{E}\left[\xi\sigma(\xi)\right], \quad \kappa_*^2 = \mathbb{E}\left[\sigma(\xi)^2\right] - \kappa_0^2 - \kappa_1^2$$
$$\kappa_0' = \mathbb{E}\left[\sigma'(\xi)\right] = \kappa_1, \quad \kappa_1' = \mathbb{E}\left[\xi\sigma'(\xi)\right], \quad (\kappa_*')^2 = \mathbb{E}\left[(\sigma'(\xi))^2\right] - (\kappa_0')^2 - (\kappa_1')^2,$$

After applying the Gaussian equivalence theorem and replica symmetry, the remaining statistical degrees of freedom are

$$\begin{cases} s_a & = \kappa_0 \langle w^*, 1_p \rangle \\ s_b & = \kappa_0' V_k^\top \Theta w^* \\ f_a & = \kappa_1 \langle \theta_0, \Theta w^* \rangle \\ f_b & = \kappa_1' V_k^\top \Theta \mathrm{diag}(w^*) \Theta^\top \theta_0 \\ \Sigma_a + q_a & = \langle w^*, M_{00} w^* \rangle \\ q_a & = \langle w^* M_{00} (w^*)' \rangle \\ \Sigma_b + q_b & = V_k^\top \Theta \mathrm{diag}(w^*) M_{01} w^* \\ q_b & = V_k^\top \Theta \mathrm{diag}(w^*) M_{01} (w^*)' \\ \Sigma_c + q_c & = V_k^\top \Theta \mathrm{diag}(w^*) M_{11} \mathrm{diag}(w^*) \Theta^\top V_k \\ q_c & = V_k^\top \Theta \mathrm{diag}(w^*) M_{11} \mathrm{diag}((w^*)') \Theta^\top V_k \end{cases}$$

The replica symmetric **fixed saddle point** system

For simplicity take $k = 1$. The core fixed point equation one needs to solve is

$$V_a = \lim_{d,n,p\to\infty} \frac{1}{p}\mathrm{tr}\left(\left(\alpha\lambda I + \hat{V}_a\Theta^\top\Theta + \hat{V}_c D_\zeta\Theta^\top\Theta D_\zeta\right)^{-1}\Theta^\top\Theta\right)$$

$$V_c = \lim_{d,n,p\to\infty} \frac{1}{p}\mathrm{tr}\left(\left(\alpha\lambda I + \hat{V}_a\Theta^\top\Theta + \hat{V}_c D_\zeta\Theta^\top\Theta D_\zeta\right)^{-1} D_\zeta\Theta^\top\Theta D_\zeta\right)$$

with $\hat{V}_a = \frac{\alpha}{1+V_a}$ and $\hat{V}_c = \frac{\alpha}{1+V_c}$, and where Θ and D_ζ are Gaussian random matrices.

The RHS are rational functions of Gaussian matrices — should have asymptotic formula using tools from **operator-valued free probability**! [FOBS06, AP20] Please talk to me later if you have expertise in this!

Conclusions

Main Takeaway: It's unclear how massively over-parametrization neural networks benefit from incorporating physical priors—here, derivative data.

Conclusions

Main Takeaway: It's unclear how massively over-parametrization neural networks benefit from incorporating physical priors—here, derivative data.

- the replica method provides one approach to systematically analyze the behaviour of Sobolev training for random feature models.
- providing gradient data does not necessarily improve generalization for RFM!
- In general, the higher the alignment ϖ, the worse the errors are.
- perhaps surprisingly, there are still regimes in which it is preferable to memorize both function and gradient noise to generalize better.

Next steps to investigate

Question: Do I actually believe that gradients help or hurt in practice?

Answer: I think it honestly really depends on the problem!

- random features cannot beat the best *linear* approximation [MMM21,GMMM21]
- deep Gaussian equivalence theorem exists, but I don't believe this meaningfully changes any conclusions [CTBB25]
- neurals are useful because they achieve **feature learning**
- recent work studies the models with (multiple) "large gradient steps" [BES+22, CPD+24, DPC+24]

Thanks for your attention, enjoy the summer! 🏄🏄🏄

End

Inserting saddle point equations just to advertise ideas

$$b = \lim_{d,n,p\to\infty} \mathbb{E}\left[\langle \hat{w}, \mathbb{1}_p \rangle\right]$$

$$q_s = \frac{1}{\gamma} \lim_{d,n,p\to\infty} \mathbb{E}\left[\|\Theta\hat{w}\|^2\right]$$

$$q_w = \lim_{d,n,p\to\infty} \mathbb{E}\left[\|\hat{w}\|^2\right]$$

$$m = \frac{1}{\sqrt{\gamma}} \lim_{d,n,p\to\infty} \mathbb{E}\left[\langle \theta_0, \Theta\hat{w} \rangle\right] ,$$

which can be found by solving the following 5-dimensional system of saddle-point equations, instead limits directly:

$$\begin{cases} V_s = \frac{1}{\hat{V}_s}\left[1 - z g_\mu\left(-z\right)\right] \\ q_s = \frac{\hat{m}^2+\hat{q}_s}{\hat{V}_s^2}\left[1 - 2z g_\mu\left(-z\right) + z^2 g'_\mu\left(-z\right)\right] - \frac{\hat{q}_w}{\left(\alpha\lambda+\hat{V}_w\right)\hat{V}_s}\left[-z g_\mu\left(-z\right) + z^2 g'_\mu\left(-z\right)\right] , \\ m = \frac{\hat{m}}{\hat{V}_s}\left[1 - z g_\mu\left(-z\right)\right] , \\ V_w = \frac{\gamma}{\alpha\lambda+\hat{V}_w}\left[\frac{1}{\gamma} - 1 + z g_\mu\left(-z\right)\right] , \\ q_w = \gamma\frac{\hat{q}_w}{\left(\alpha\lambda+\hat{V}_w\right)^2}\left[\frac{1}{\gamma} - 1 + z^2 g'_\mu\left(-z\right)\right] - \gamma\frac{\hat{m}^2+\hat{q}_s}{\left(\alpha\lambda+\hat{V}_w\right)\hat{V}_s}\left[-z g_\mu\left(-z\right) + z^2 g'_\mu\left(-z\right)\right] , \end{cases}$$

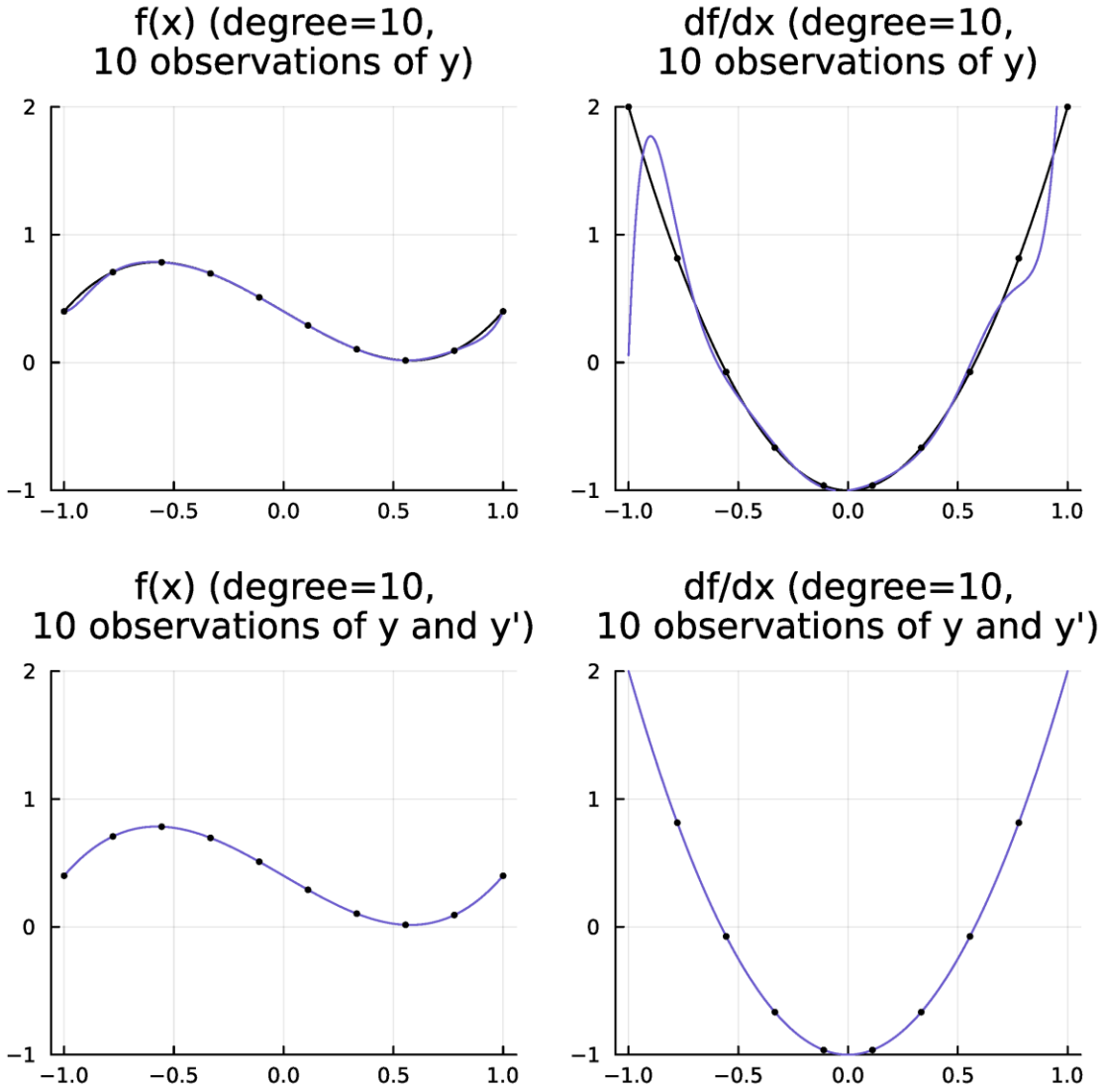
$_s$, where the auxiliary variables with hats are defined as

$$\begin{cases} \hat{V}_s = \frac{\alpha\kappa_1^2}{\gamma V_*}\mathbb{E}_\xi\left[\int_{\mathbb{R}} \mathcal{Z}[P]\left(y;\bar{m}_1,\sigma_1^2\right)\cdot\left(1 - \partial_{\bar{m}_2}\tilde{y}_\ell^*\left(y;\bar{m}_2,\sigma_2^2\right)\right)\mathrm{d}y\right] , \\ \hat{q}_s = \frac{\alpha\kappa_1^2}{\gamma V_*^2}\mathbb{E}_\xi\left[\int_{\mathbb{R}} \mathcal{Z}[P]\left(y;\bar{m}_1,\sigma_1^2\right)\cdot\left(\tilde{y}_\ell^*\left(y;\bar{m}_2,\sigma_2^2\right) - \bar{m}_2\right)^2\mathrm{d}y\right] , \\ \hat{m} = \frac{\alpha\kappa_1}{\gamma V_*}\mathbb{E}_\xi\left[\int_{\mathbb{R}} \partial_{\bar{m}_1}\mathcal{Z}[P]\left(y;\bar{m}_1,\sigma_1^2\right)\cdot\left(\tilde{y}_\ell^*\left(y;\bar{m}_2,\sigma_2^2\right) - \bar{m}_2\right)\mathrm{d}y\right] , \\ \hat{V}_w = \frac{\alpha\kappa_*^2}{V_*}\mathbb{E}_\xi\left[\int_{\mathbb{R}} \mathcal{Z}[P]\left(y;\bar{m}_1,\sigma_1^2\right)\cdot\left(1 - \partial_{\bar{m}_2}\tilde{y}_\ell^*\left(y;\bar{m}_2,\sigma_2^2\right)\right)\mathrm{d}y\right] , \\ \hat{q}_w = \frac{\alpha\kappa_*^2}{V_*^2}\mathbb{E}_\xi\left[\int_{\mathbb{R}} \mathcal{Z}[P]\left(y;\bar{m}_1,\sigma_1^2\right)\cdot\left(\tilde{y}_\ell^*\left(y;\bar{m}_2,\sigma_2^2\right) - \bar{m}_2\right)^2\mathrm{d}y\right] . \end{cases}$$

$$\tilde{y}_\ell^*\left(y;\bar{m}_2,\sigma_2^2\right) = \operatorname*{arg\,min}_{\tilde{y}\in\mathbb{R}}\left\{\ell\left(y-\tilde{y}\right) + \frac{\left(\tilde{y}-\bar{m}_2\right)^2}{2\sigma_2^2}\right\}$$

$$\mathcal{Z}[P]\left(y;\bar{m}_1,\sigma_1^2\right) = \mathbb{E}_{\omega\sim\mathcal{N}\left(\bar{m}_1,\sigma_1^2\right)}\left[P\left(y \mid \omega\right)\right] ,$$

Interlude: Sobolev training with **polynomial features**



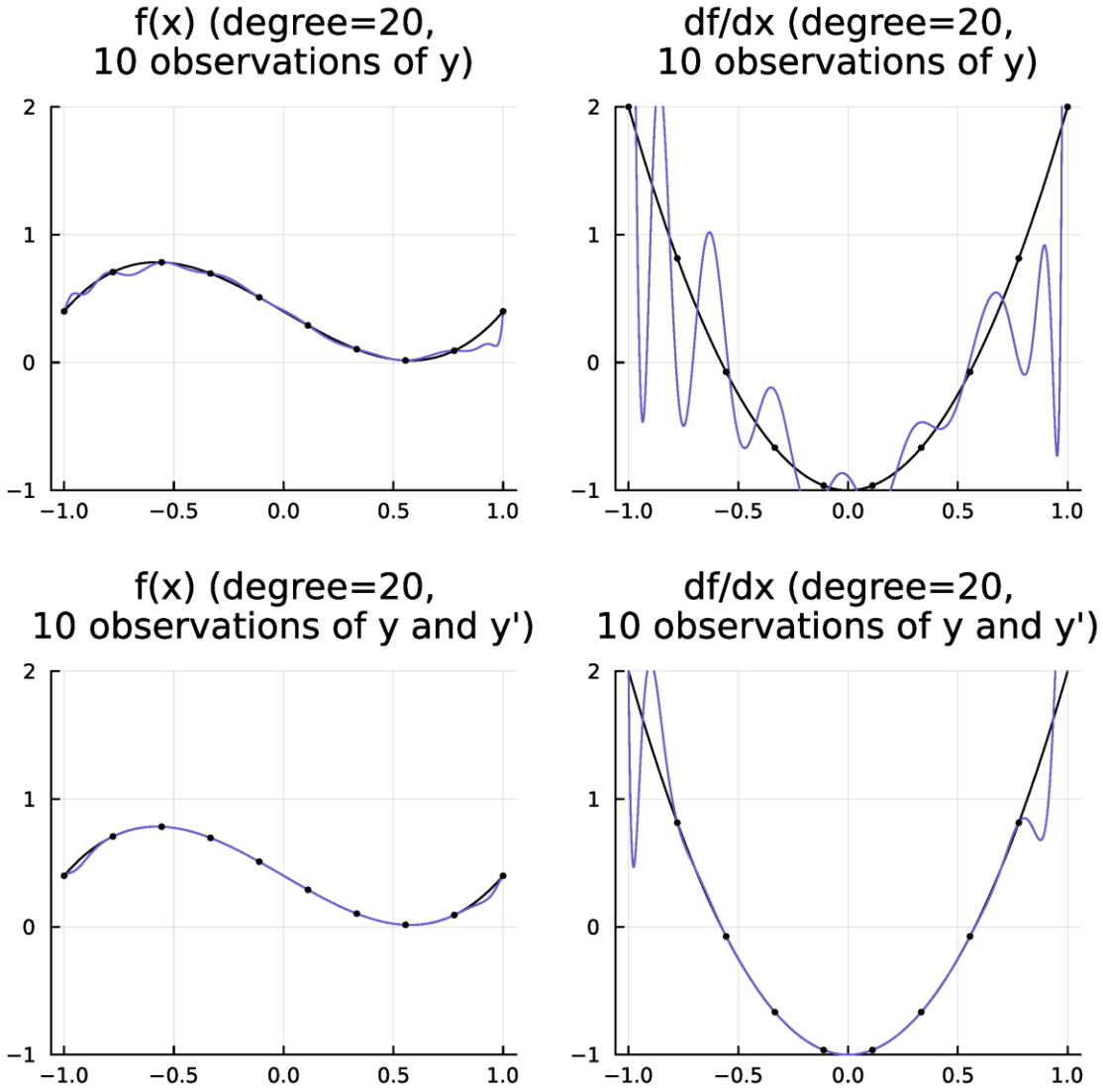
Let $\begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$

and find the **minimum norm interpolant**

$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

Note: figures are similar even with noise-less data.

Interlude: Sobolev training with **polynomial features**



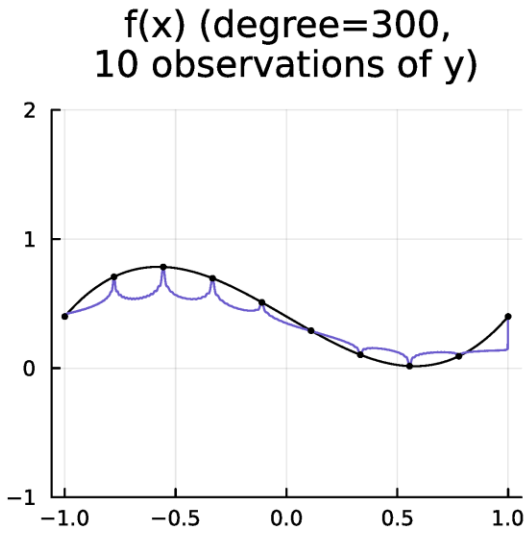
Let $\begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$

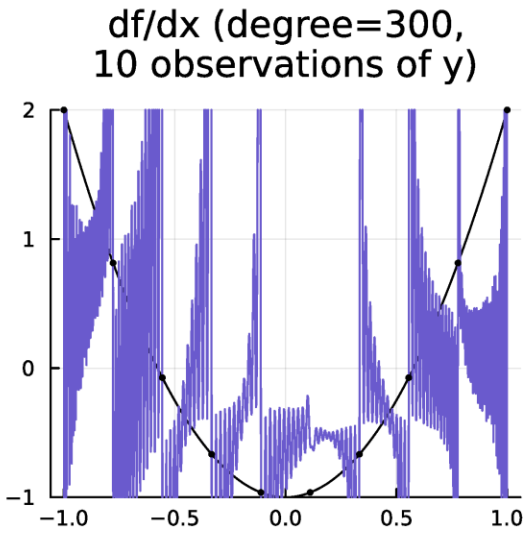
and find the **minimum norm interpolant**

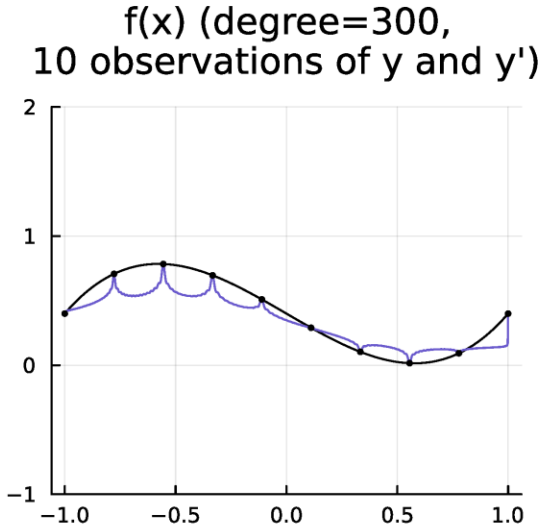
$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

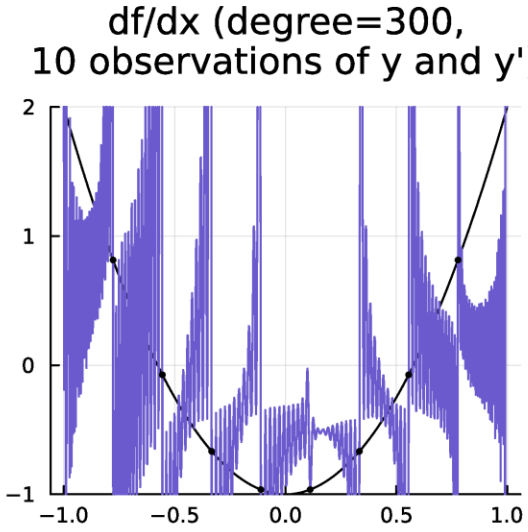
Note: figures are similar even with noise-less data.

Interlude: Sobolev training with **polynomial features**









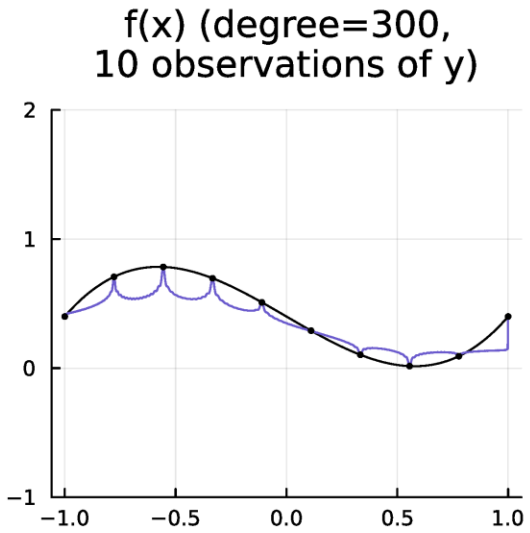
Let $\begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$

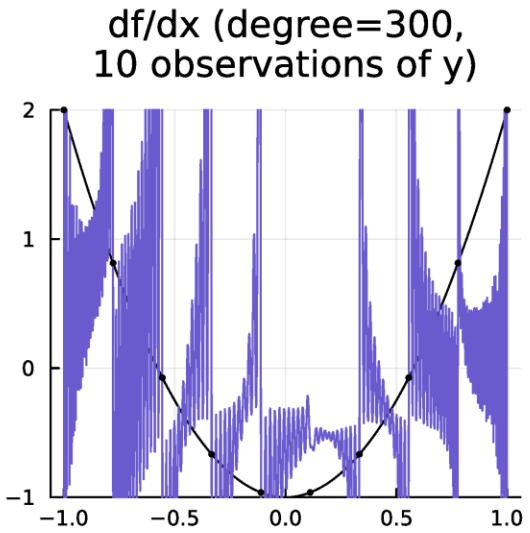
and find the **minimum norm interpolant**

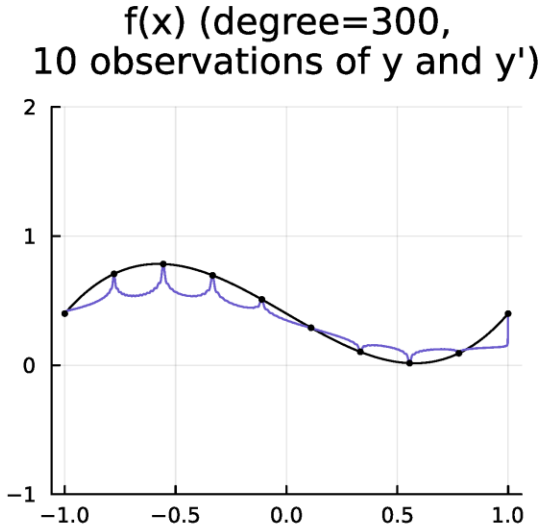
$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

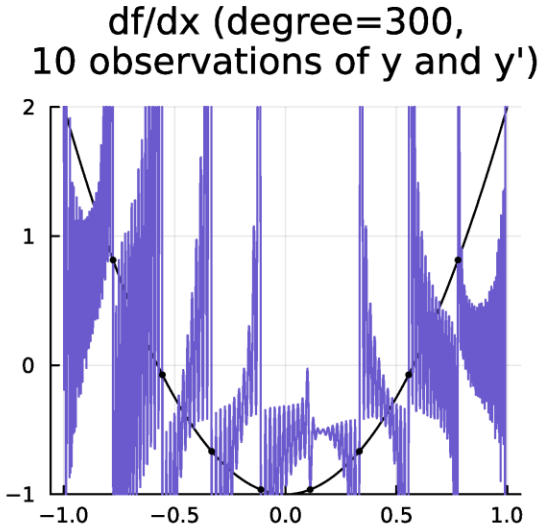
Note: figures are similar even with noise-less data.

Interlude: Sobolev training with **polynomial features**









Let $\begin{cases} \Psi(x) = \begin{bmatrix} \varphi_0(x) & \varphi_1(x) & \dots & \varphi_p(x) \end{bmatrix} \\ \Psi'(x) = \begin{bmatrix} \varphi_0'(x) & \varphi_1'(x) & \dots & \varphi_p'(x) \end{bmatrix} \end{cases}$

and find the **minimum norm interpolant**

$$\min_{w \in \mathbb{R}^p} \|w\|_2 \quad s.t. \quad \begin{pmatrix} y \\ y' \end{pmatrix} = \begin{pmatrix} \Psi(x) \\ \Psi'(x) \end{pmatrix} w$$

Note: figures are similar even with noise-less data.