Generalization by Weight Decay and Large Learning Rate in "curve fitting" with *neural networks*

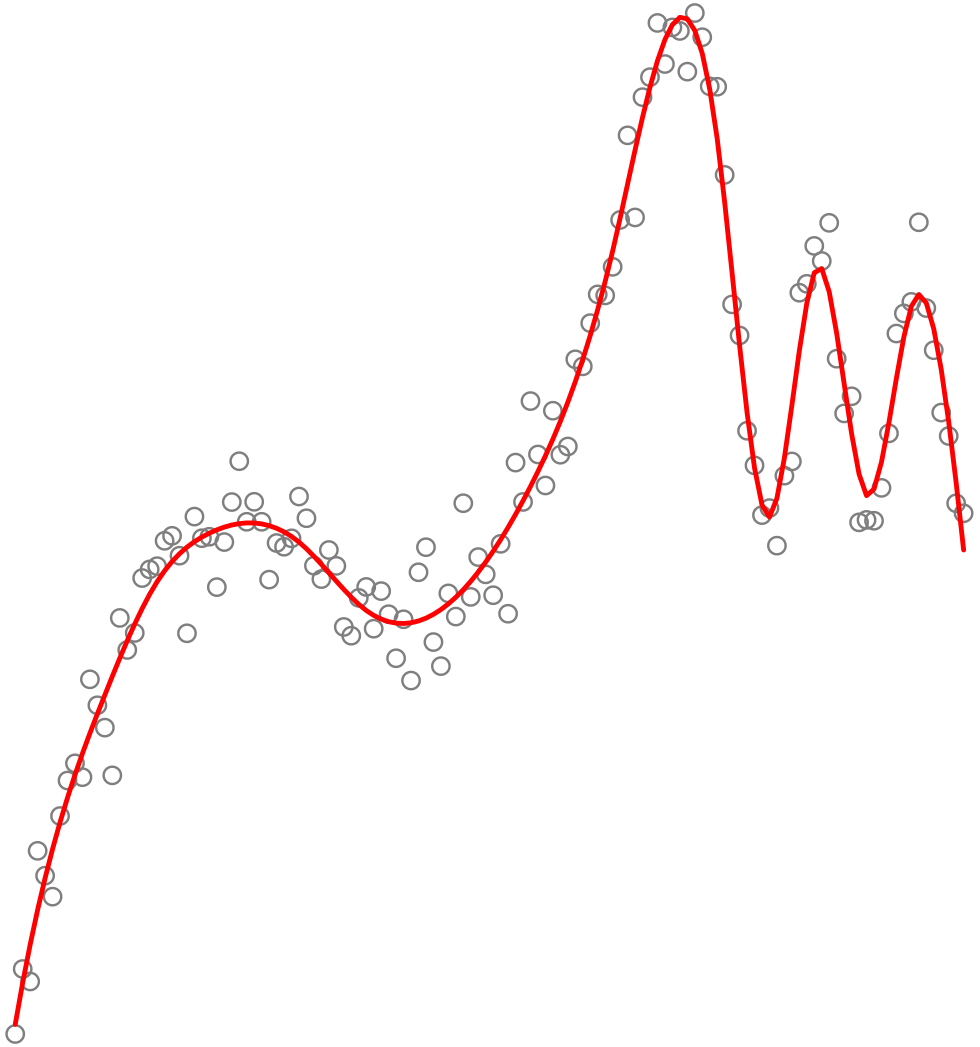
Yu-Xiang Wang

Based on joint work with Kaiqi Zhang, Dan Qiao, Esha Singh, Zixuan Zhang, Minshuo Chen, Tuo Zhao and Daniel Soudry

COMPUTER SCIENCE
UC SANTA BARBARA
Computing. ReInvented.

UC San Diego
HALICIOĞLU DATA SCIENCE INSTITUTE

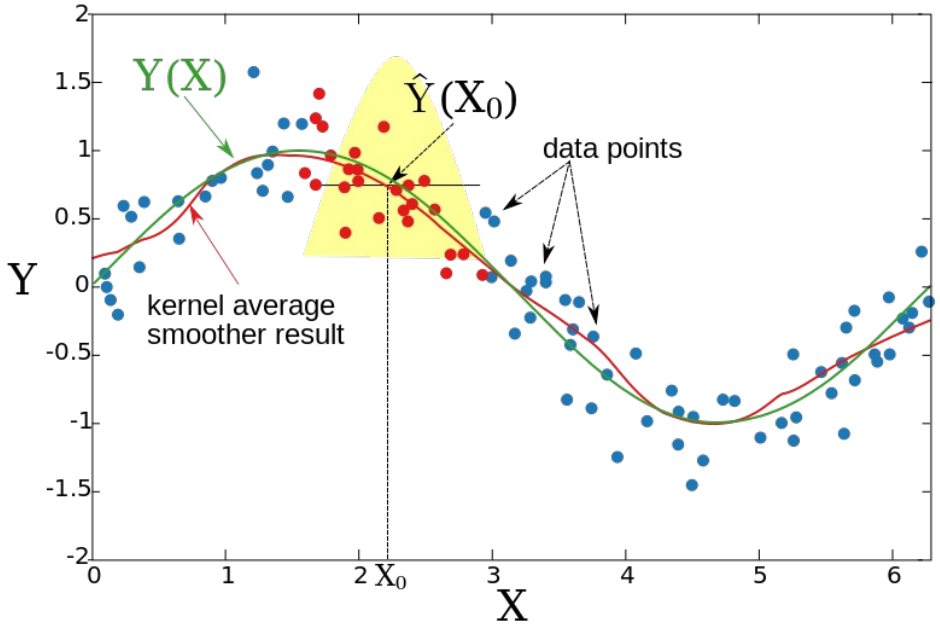
“Is deep learning is just curve fitting? ”



Judea Pearl
Turing Award Laureate
2011

Curve fitting isn't easy

- 60 years of associated literature
  - Since [Nadaraya, Watson, 1964] then Wahba, Donoho, Mammen, Van de Geer, Barron, Bartlett, Schölkopf etc…
  - **Methods:** Kernels, splines, local polynomials, Gaussian processes and RKHS, CART, MARS, neural networks



a.k.a., smoothing / filtering / denoising / signal processing / nonparametric regression.

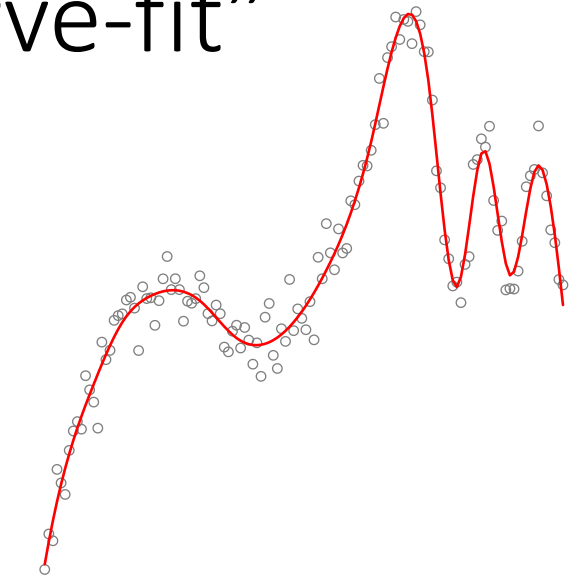
Not whether you can "curve-fit" but *how well (and how efficiently)* you can "curve-fit"

$$y_i = f(x_i) + \text{Noise for } i = 1, ..., n.$$

Goal: Estimate the function using noisy data

$$(x_1, y_1), ..., (x_n, y_n)$$

assume that $f \in \mathcal{F}$



$$\text{MSE}(\hat{f}) := \mathbb{E}_{\mathcal{D}_n} \frac{1}{n} \sum_{i=1}^{n} (\hat{f}(\boldsymbol{x}_i) - f_0(\boldsymbol{x}_i))^2.$$

- Approximation theory
- Estimation (generalization) theory
- Optimality

“Curve fitting” is a perfect test-bed for understanding the fundamental properties of deep learning

- What functions do DNNs represent?
- Overparameterization
- How does training / regularization affect generalization?

Why do Neural Networks work better?

- Universal function approximation (Hornik et al, 1989)
  - But so are kernels and splines!
- Flexible representation and modelling language
  - So are graphical models / probabilistic programs
- Overparameterization
  - Neural Tangent Kernels (Jacot et al., 2018; Du et al. 2019; etc)
  - Interpolation regime / benign overfitting (Belkin, 2019, Bartlett et al. 2020, Frei, 2022 and more)

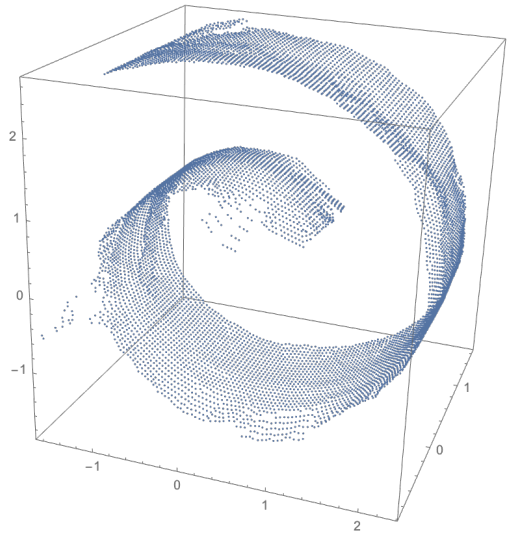
The “adaptivity” hypothesis

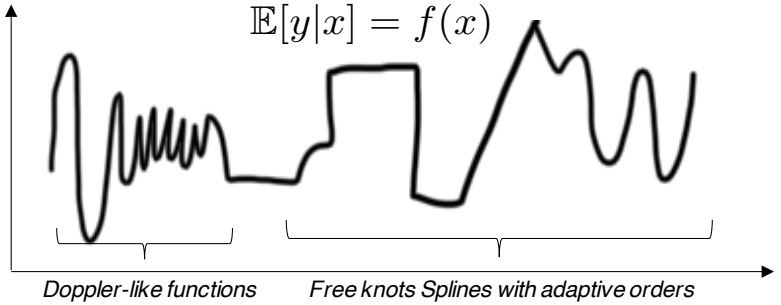
- Neural networks aren’t stronger than classical methods in any specific problem

- But the standard practices of how people develop / train **DNNs enjoy strong adaptivity**
  - No need to carefully specify the problem
  - Automatically choose the right level of abstraction
  - Tune only standard hyperparameters.
  - They match the best classical methods on each problem

Recent work from my group sheds light on how DNNs are adaptive.

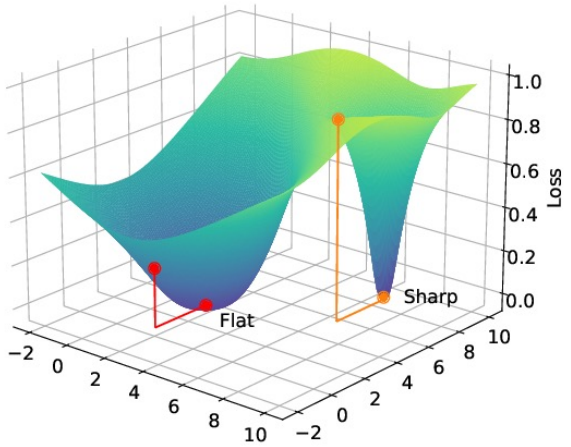
1. "Are Weight Decayed DNNs locally adaptive?"



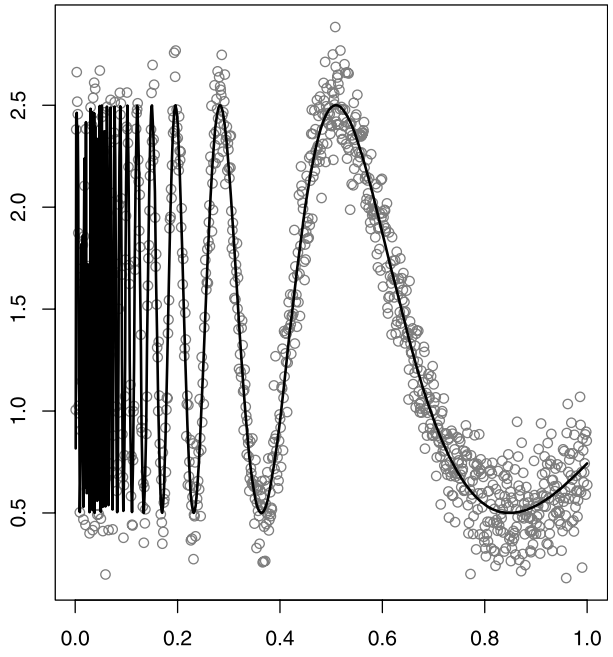


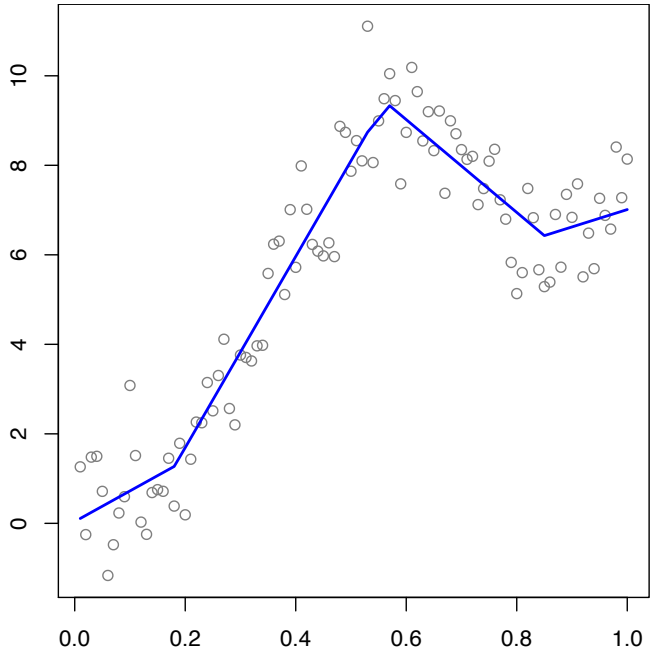
2. "Weight Decayed DNNs adapt to low-dimensional manifolds" (will skip)

3. Generalization by Large Learning Rate in Gradient Descent



Locally adaptive nonparametric regression





- Some parts smooth, other parts wiggly.
  - Wavelets [Donoho&Johnston,1998], adaptive kernel [Lepski,1999], adaptive splines [Mammen&Van De Geer,2001], Trend filtering [Steidl,2006; Kim et. al. 2009, Tibshirani, 2013; W., Smola, Tibshirani, 2014], adaptive online local polynomials [Baby and W., 2018/19]
  - a.k.a, multiscale, multi-resolution compression, used in JPEG2000.

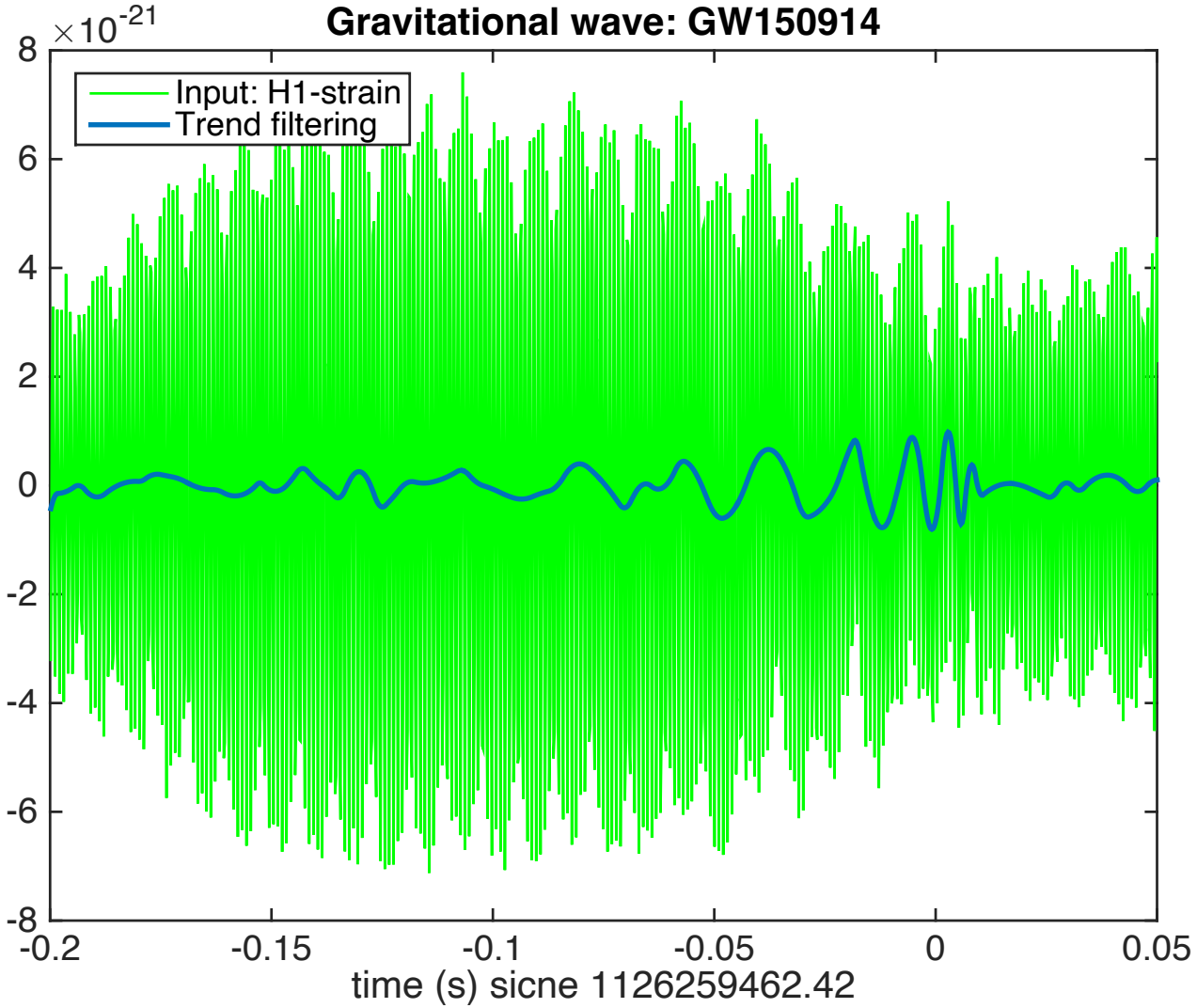
NTKs are strictly suboptimal for locally adaptive nonparametric regression

- Observations: $y_i = f_0(x_i) + \epsilon_i, \quad i = 1, \ldots n$
- TV-class: $\mathcal{F}_m = \left\{ f : \mathrm{TV}(f^{(m)}) \leq C \right\}$
- Minimax error rate: $O_{\mathbb{P}}(n^{-(2m+2)/(2m+3)})$
- Best achievable rate for linear smoothers (e.g., any **kernel ridge regression, including NTK.**)

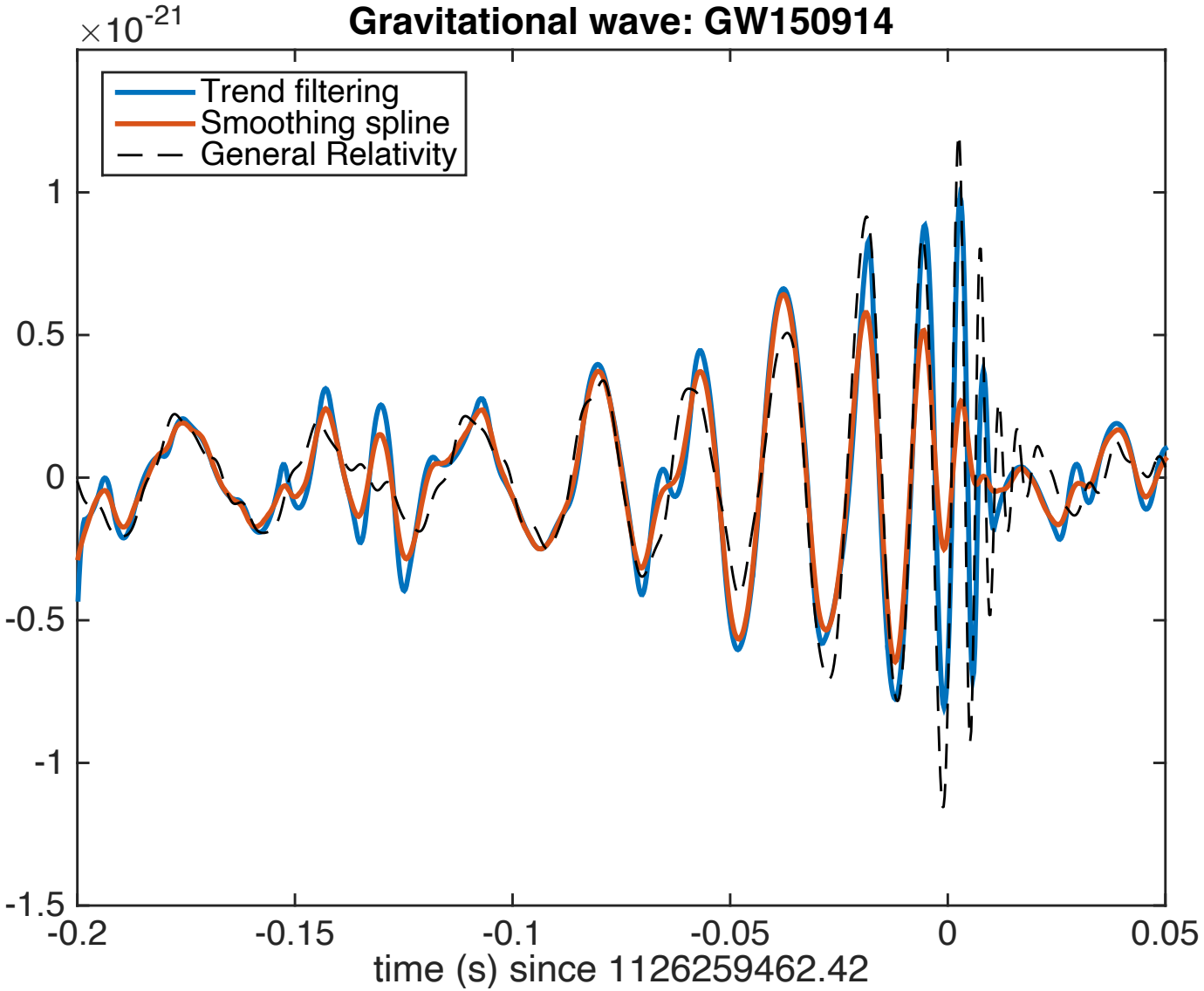
$$O_{\mathbb{P}}(n^{-(2m+1)/(2m+2)})$$

(Tibshirani, 2014, Annals of Statistics) (Donoho, Liu, MacGibbon, 1990)

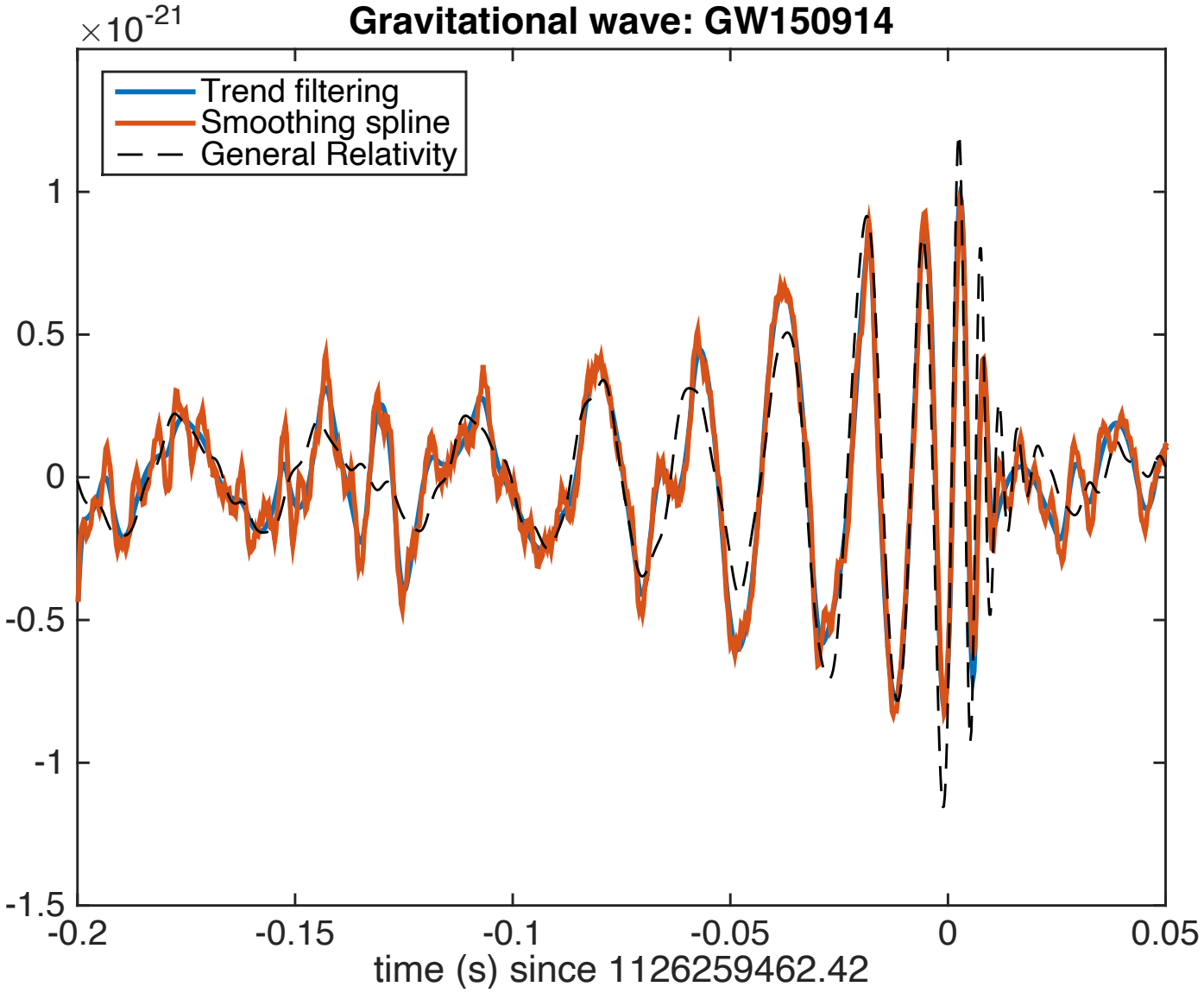
Example: merger of two black holes



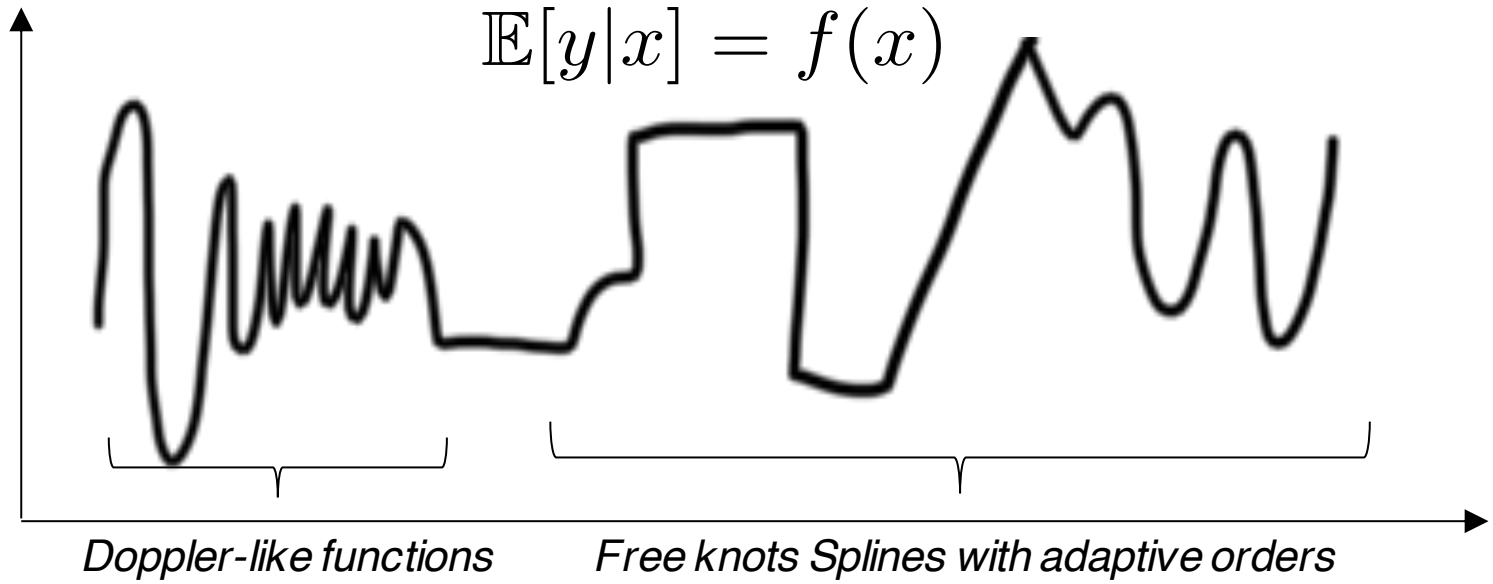
Example: merger of two black holes



Example: merger of two black holes



Are DNNs locally adaptive? Can they achieve optimal rates for TV-classes / Besov classes?



Are DNNs locally adaptive? Can they achieve optimal rates for TV-classes / Besov classes?

- Existing work:
  - Suzuki (2019): Specific ReLU NN achieves minimax rate for Besov classes. (albeit with width, depth, sparsity constraints tailored to each problem)
  - Liu, Chen, Zhao, Liao (2021): ConvResNets works too. No sparsity, but similarly requires the number of parameters to be small.
  - Parhi and Nowak (2021): 2-layer NN is equivalent to Locally Adaptive Regression Splines (LAR Splines)

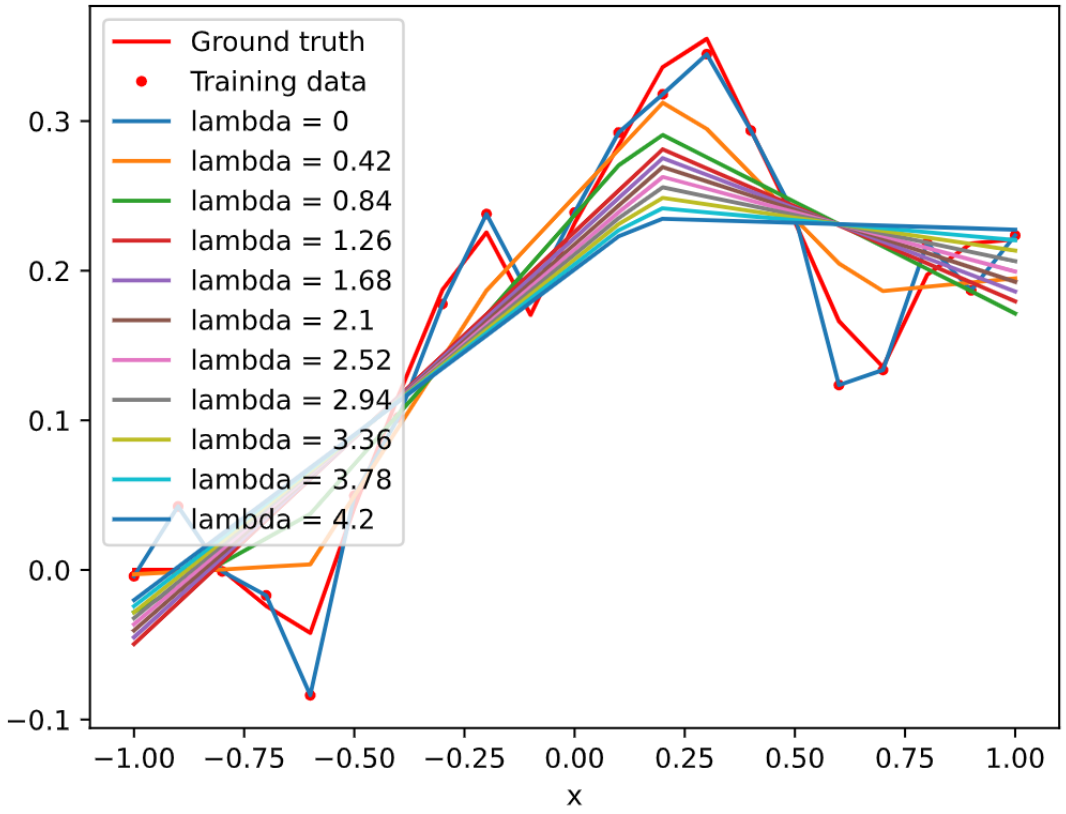
Our results (Zhang and W. 2023): Parallel Deep NN achieves **near-optimal local adaptive rates**, simultaneously for many classes

- Tuning only weight decay / no architecture search.
- Depth is important. Implicit sparsity solves both representation learning and overparameterization.

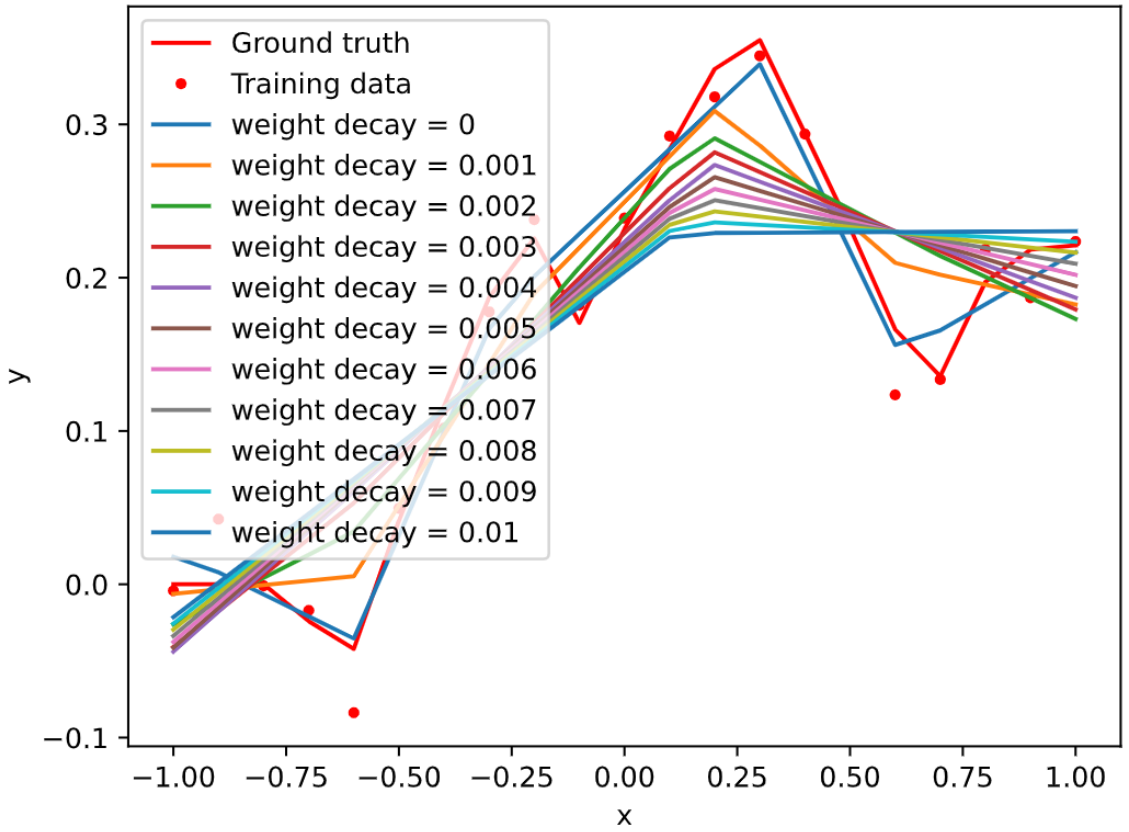
**Disclaimer: We ignore computation and focus on understanding the statistical property of the ERM.*

Let's first look at an experiment: compare ReLU NN with L1 trend filtering

L1-Trend Filtering

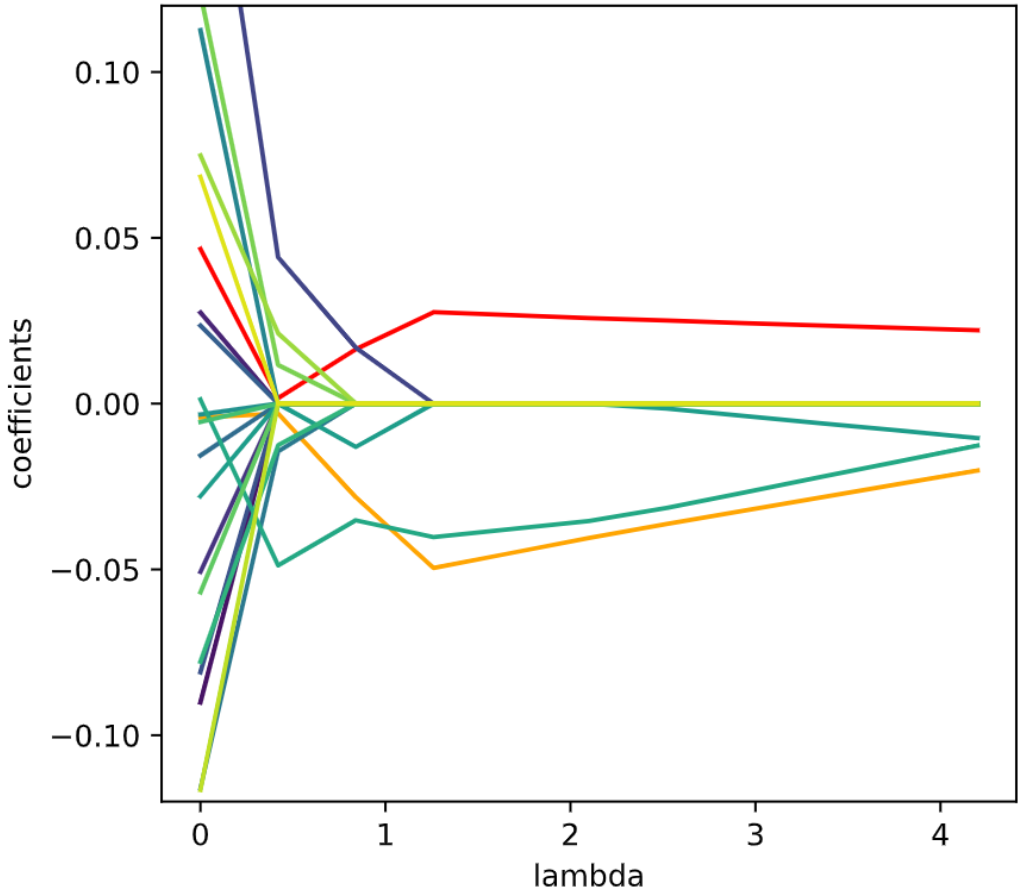


Two-layer ReLU NN with weight decay

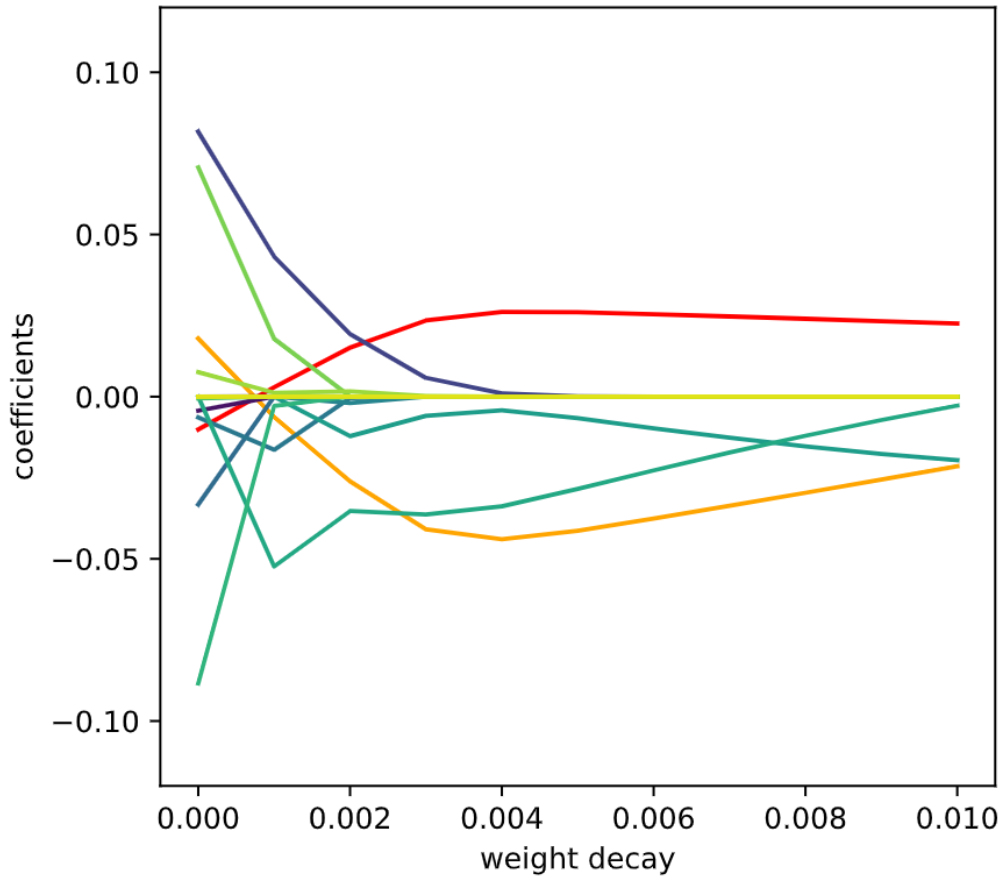


Let’s inspect the regularization paths

L1-Trend Filtering



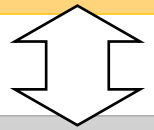
Two-layer ReLU NN with weight decay



It turns out that they are equivalent!

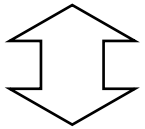
L1-Trend Filtering

(Kim, Koh and Boyd, 2009; Tibshirani, 2024)



LAR Splines

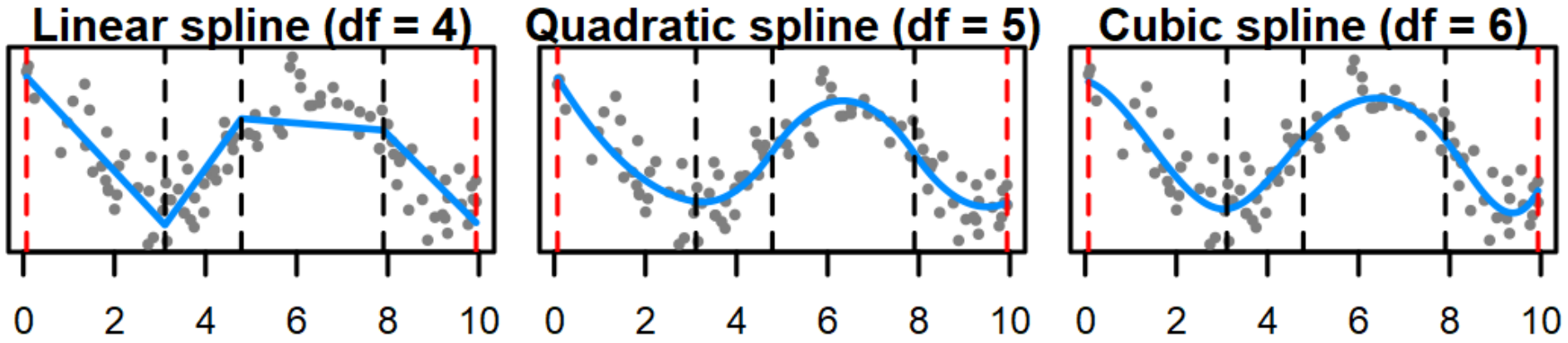
(Mammen and Van De Geer, 1997)



Reveals the effect of weight decay

(Sufficiently wide) Two-layer ReLU Neural Networks with Weight Decay

Background: Splines are piecewise polynomials



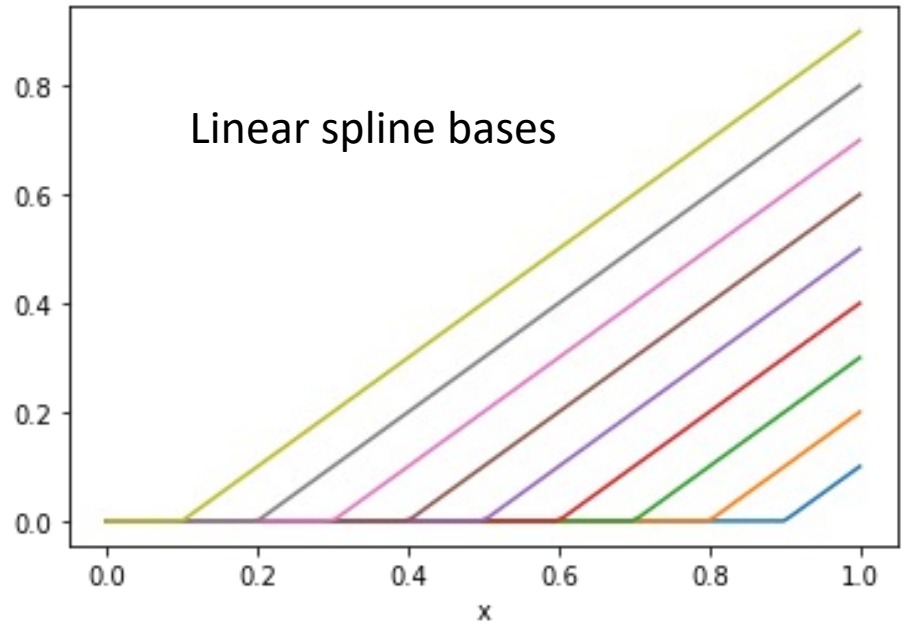
(Illustration from a Stats.Stackexchange contributor)

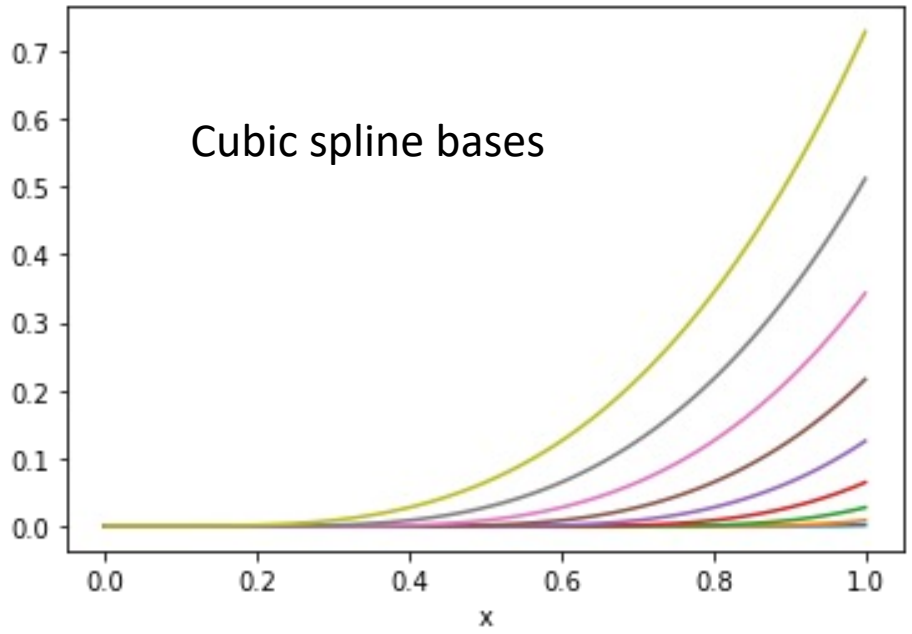
- Where to choose knots?
  - **Smoothing splines:** choose n of them, one on each input data point and do L2 penalty on the coefficients
  - **LAR splines:** select a **sparse number** of them using L1-penalty.

Background: Truncated power basis for splines

- Pick knots at $t_1, \ldots, t_M$
- A set of basis functions that spans the splines

$$1, x, x^2, \ldots, x^m, (x - t_1)_+^m, \ldots, (x - t_M)_+^m$$





Observation: Two-layer NNs **are** ~~approximating~~ Free-Knot Splines

- Neural networks $$f(x) = \sum_{j=1}^{M} v_j \sigma^m(w_j x + b_j) + c(x),$$

- Splines / truncated power-basis

$$f(x) = \sum_{j=1}^{M} c_j \sigma^m(x - t_j) + \tilde{c}(x)$$

- Only difference
  - Trend filtering / smoothing splines fixed the knots at input data points
  - NN left them freely moving, i.e., free-knot splines (Jupp 1978; Kass et al. 2001)

Weight decay = Total Variation Regularization

- Neural networks

$$f(x) = \sum_{j=1}^{M} v_j \sigma^m(w_j x + b_j) + c(x),$$
$$= \sum_{j=1}^{M} c_j \sigma^m(x - t_j) + \tilde{c}(x)$$

- Weight decay

$$\min_{\boldsymbol{w},\boldsymbol{v}} \hat{L}(f) + \frac{\lambda}{2} \sum_{j=1}^{M} (|v_j|^2 + |w_j|^{2m}) = \lambda \sum_j |c_j| = \mathrm{TV}(f^{(m)})$$

At the optimal solutions

- AM-GM inequality

$$|v_j|^2 + |w_j|^{2m} \geq 2|v_j||w_j|^m = 2|c_j|$$

  - Observed by (Neyshabur et al., 2014), (Parhi and Nowak, 2021), (Tibshirani, 2021) etc…

Two-layer Weight-Decayed NN is equivalent to LAR Splines (Parhi and Nowak, 2021) when mildly overparameterized

- When the number of knots M > n - m
  - Banach space representer Thm (Theorem 8 of Parhi and Nowak, 2021)

$$\min_{\boldsymbol{w},\boldsymbol{v}} \hat{L}(f) + \frac{\lambda}{2}\sum_{j=1}^{M}(|v_j|^2 + |w_j|^{2m}) \quad \Leftrightarrow \quad \min_{f} \hat{L}(f) + \lambda TV(f^{(m)}(x)),$$

over all functions!

- By Mammen and Van De Geer (1997)

$$\mathrm{MSE}(\hat{f}) = O(n^{-(2m+2)(2m+3)}). \quad \textbf{(Minimax rate)}$$

Checkpoint: Overparameterize + Weight Decay in 2-layer NN is an optimal smoother for TV1.

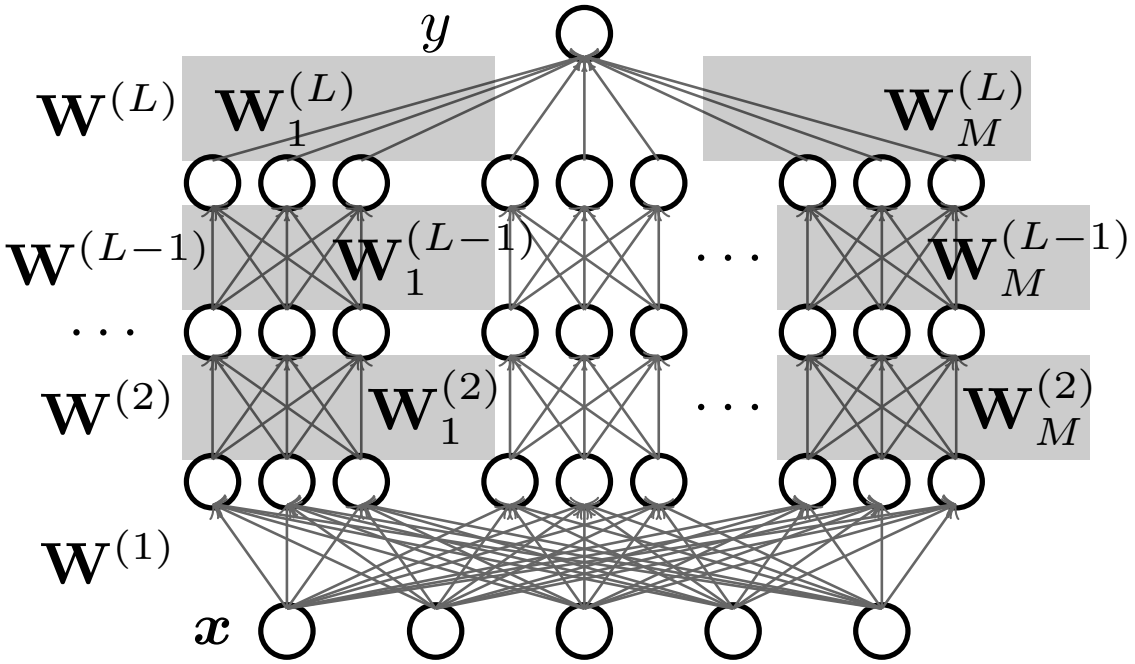
- Tuning weight decay is equivalent to tuning the TV1 radius.

- Explains what kernels cannot explain! are provably suboptimal.

- Limitations:
  - Effect of depth?
  - Adaptivity to smoother classes? TV2, TV3, etc…

Can we generalize the result?

L-Layer *Parallel* Neural Networks

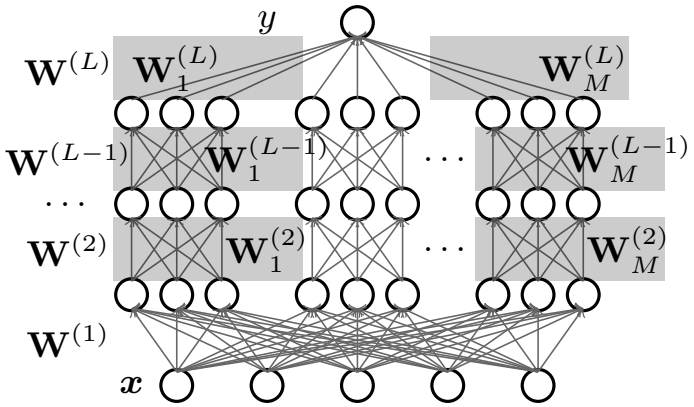


$$\min_{f_j} L(\textstyle\sum_j f_j) + \lambda \sum_{\ell=1}^{L} \sum_{j=1}^{M} \|\mathbf{W}_j^{(\ell)}\|_F^2.$$

(a) Parallel NN with Weight Decay

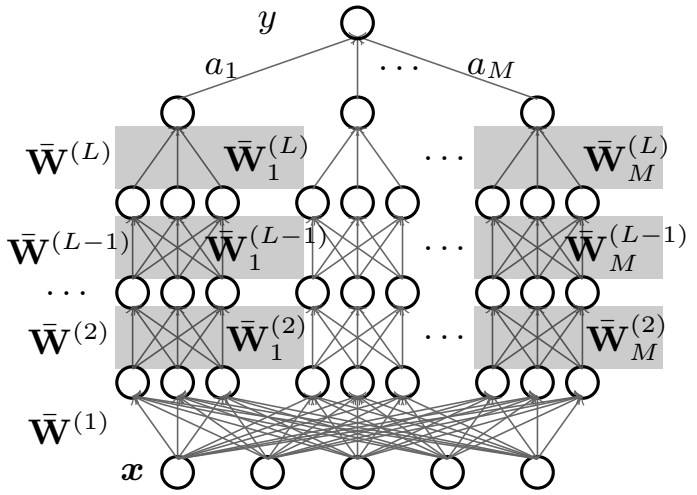
(Ergen&Pilanci, 2021; Haeffele & Vidal, 2017). Also, SqueezeNet, ResNeXT etc.

Weight decayed L-Layer PNN is equivalent to Sparse Linear Regression with learned basis functions



$\min_{f_j} L(\sum_j f_j) + \lambda \sum_{\ell=1}^{L} \sum_{j=1}^{M} \|\mathbf{W}_j^{(\ell)}\|_F^2.$

(a) Parallel NN with Weight Decay



$\min_{\{a_j, \bar{f}_j\}} L(\sum_j a_j \bar{f}_j)\ s.t. \sum_{j=1}^{M} |a_j|^{2/L} \le P'.$

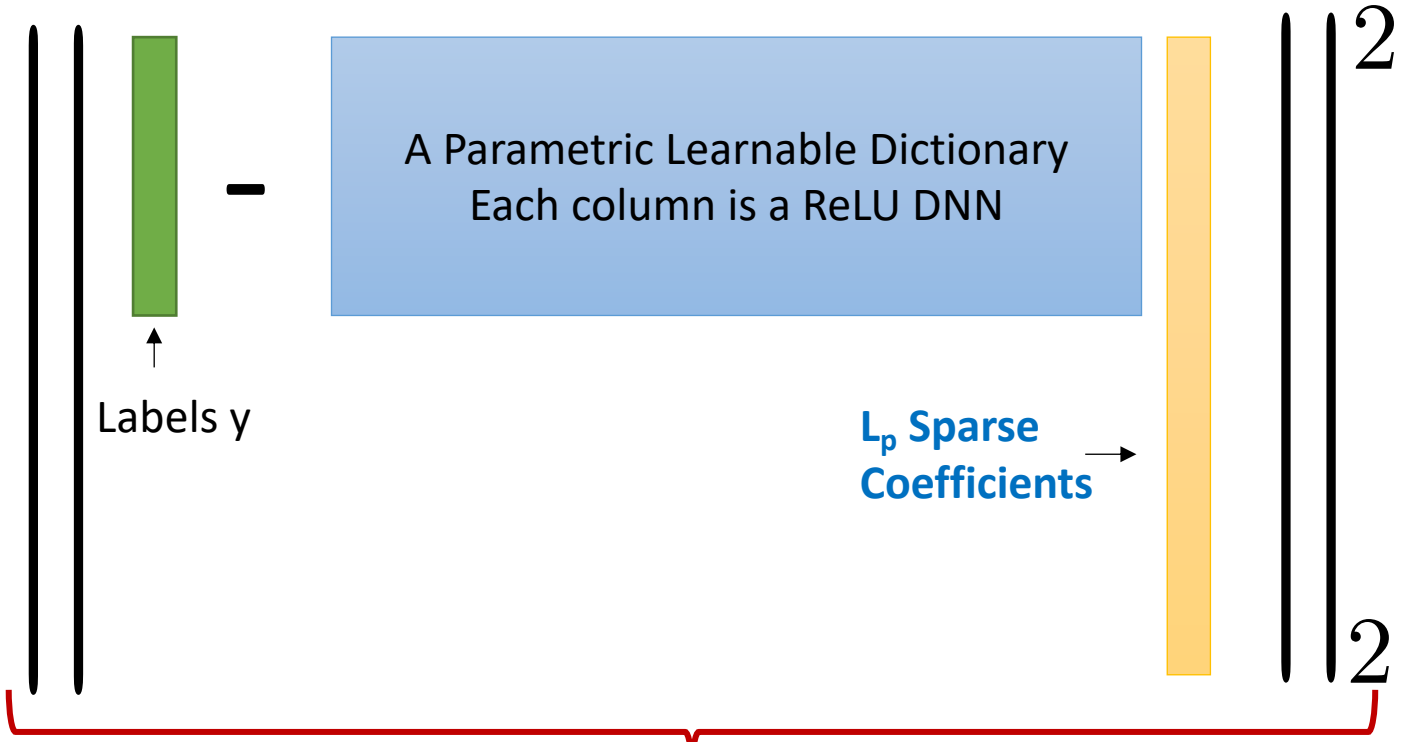
(b) Sparse Regression with Learned Representation

$$\arg\min_{\{\bar{\mathbf{W}}_j^{(\ell)}, \bar{\boldsymbol{b}}_j^{(\ell)}, a_j\}} \hat{L}\Big(\sum_{j=1}^{M} a_j \bar{f}_j\Big) = \frac{1}{n}\sum_i (y_i - \bar{f}_{1:M}(\boldsymbol{x}_i)^T \boldsymbol{a})^2$$

$$s.t.\ \|\bar{\mathbf{W}}_j^{(1)}\|_F \le c_1\sqrt{d}, \forall j \in [M],$$

$$\|\bar{\mathbf{W}}_j^{(\ell)}\|_F \le c_1\sqrt{w}, \forall j \in [M], 2 \le \ell \le L, \qquad \|\{a_j\}\|_{2/L}^{2/L} \le P'$$

Weight decayed L-Layer PNN is equivalent to Sparse Linear Regression with learned basis functions



$$\arg\min_{\{\bar{\mathbf{W}}_j^{(\ell)},\bar{\boldsymbol{b}}_j^{(\ell)},a_j\}} \hat{L}\Big(\sum_{j=1}^{M} a_j \bar{f}_j\Big) = \frac{1}{n}\sum_i (y_i - \bar{f}_{1:M}(\boldsymbol{x}_i)^T \boldsymbol{a})^2$$

$$s.t.\ \|\bar{\mathbf{W}}_j^{(1)}\|_F \le c_1\sqrt{d}, \forall j \in [M],$$

$$\|\bar{\mathbf{W}}_j^{(\ell)}\|_F \le c_1\sqrt{w}, \forall j \in [M], 2 \le \ell \le L, \quad \|\{a_j\}\|_{2/L}^{2/L} \le P'$$

Formal setup / notations

- Function classes
  - Bounded Variation class: $BV(m) := \left\{ f \in L_\infty : TV(f^{(m)}) < \infty \right\}$
  - Besov class $B^{\alpha}_{p,q}$ d-dimensional
  - Connections: $B^{m+1}_{1,1} \subset BV(m) \subset B^{m+1}_{1,\infty}$
- Metric $$\mathrm{MSE}(\hat{f}) := \mathbb{E}_{\mathcal{D}_n} \frac{1}{n} \sum_{i=1}^{n} (\hat{f}(\boldsymbol{x}_i) - f_0(\boldsymbol{x}_i))^2.$$
- Problem setting:
  - Fixed design, subgaussian noise

Main theorem: Parallel ReLU DNN approaches the minimax rates as it gets deeper.

	Minimax Rate	Minimax Linear Rate
Besov Space	$n^{-\frac{2\alpha}{2\alpha+d}}$	$n^{-\frac{2\alpha-1}{2\alpha+d-1}}$
Bounded Variation	$n^{-\frac{2m+2}{2m+3}}$	$n^{-\frac{2m+1}{2m+2}}$

- **Theorem 2:** Besov space $B^{\alpha}_{p,q}$

$$\mathrm{MSE}(\hat{f}) = \tilde{O}\left(n^{-\frac{2\alpha/d(1-2/L)}{2\alpha/d+1-2/(pL)}}\right) + O(e^{-c_6 L})$$

- **Corollary 3** for $BV(m)$ class:

$$\mathrm{MSE}(\hat{f}) = \tilde{O}(n^{-\frac{(2m+2)(1-2/L)}{2m+3-2/L}}) + O(e^{-c_6 L}),$$

Arbitrarily close to the minimax rates when we choose L = C log n.

Many interesting insights we can read off from the theorem

1. Formal separation from kernels (NTK or other kernel ridge regressions)
   - Our upper bound + Donoho, Liu, MacGibbon (1990)'s linear smoother lower bound.

2. Deep NNs achieve smaller error than shallow NNs

3. Overparameterization does not cause overfitting
   - Due to explicit regularization (i.e., weight decay) => sparsity

Comparing to classical nonparametric regression methods

$$\hat{f}(x) = \sum_{i=1}^{M} g_i(x) c_i$$

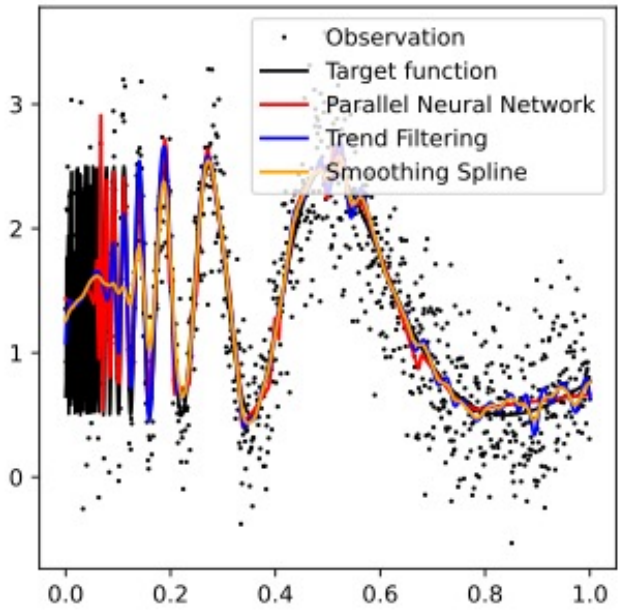
$$[g_1, \ldots, g_M]$$

$$c_{1:M} \in \mathbb{R}^M$$

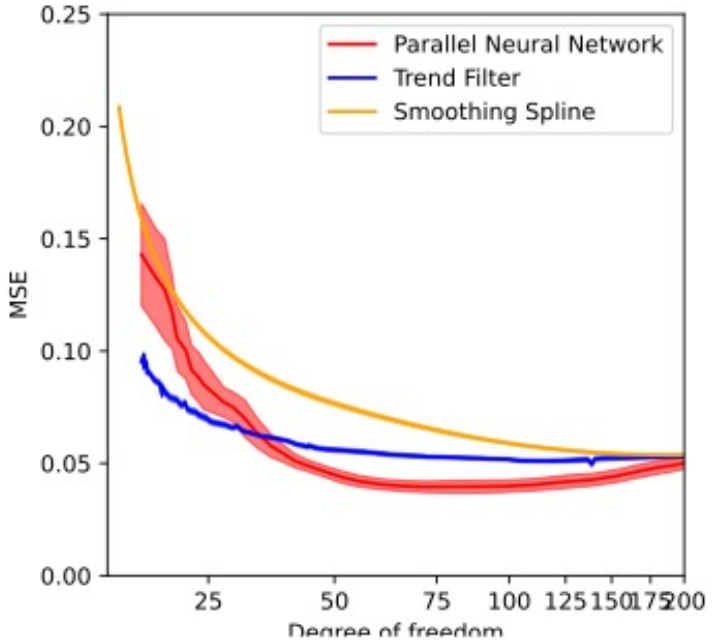
	LAR Splines / Trend filtering	Wavelet smoothing	Parallel DNN
Basis functions	Hard-coded for each order of smoothness	Hard-coded to the chosen wavelets	Parametric and learned from data.
Coefficient vector	L1-sparsity	L1 or L0-sparsity	Lp sparsity (p=2/L)

- DNNs adapt to different function classes
  - By overparameterizing / learning representation and tuning regularization weight via cross-validation (implicitly selecting a few basis functions!)
  - Paying almost no statistical price!

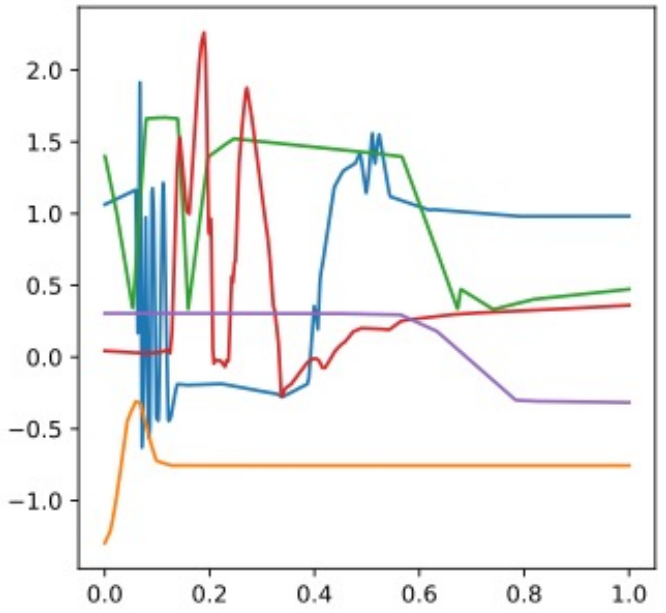
Examples of Functions with Heterogeneous Smoothness



Fitted functions with optimally tuned parameter



MSE comparison over effective degree of freedom

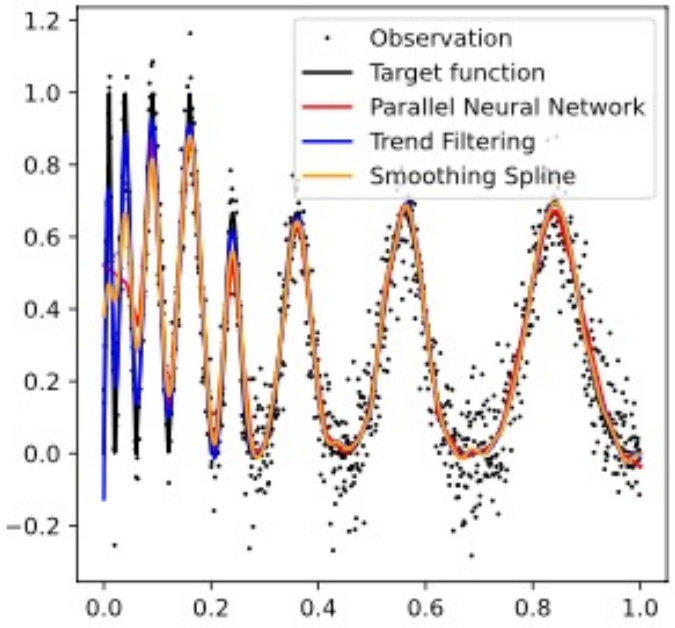


Learned basis functions. Only a handful that are active, i.e. sparsity. Lottery ticket?

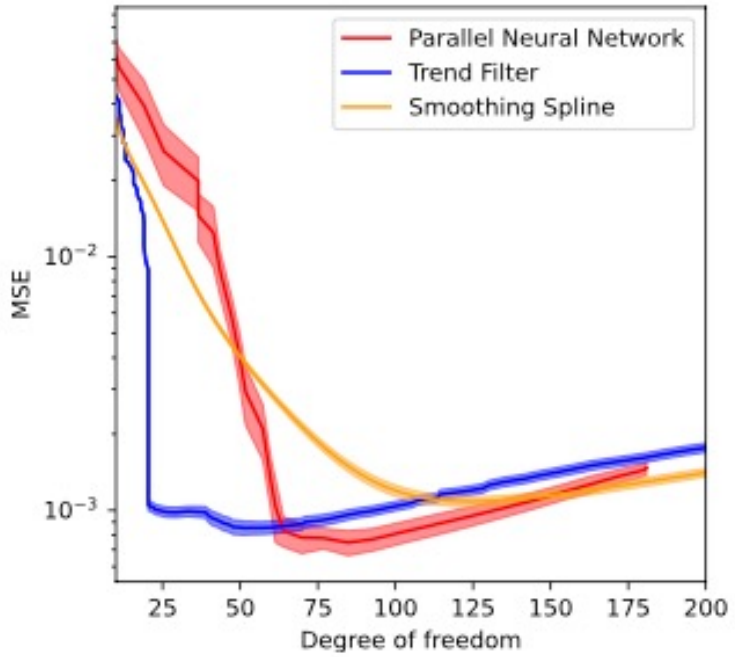
Examples of Functions with even more Heterogeneous Smoothness

Piecewise Linear

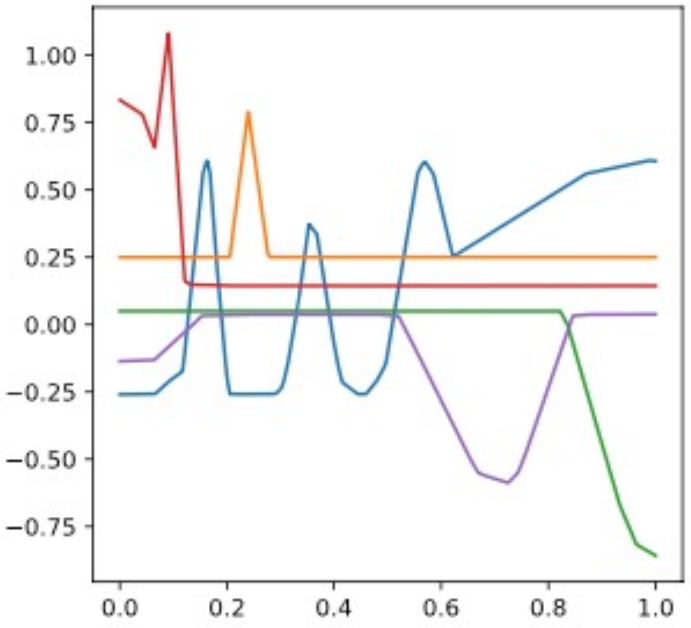
Piecewise Cubic



Fitted functions with optimally tuned parameter



MSE comparison over effective degree of freedom



Learned basis functions. Only a handful that are active, i.e. sparsity. Lottery ticket?

Checkpoint: PNN with weight decay enjoys amazing adaptivity.

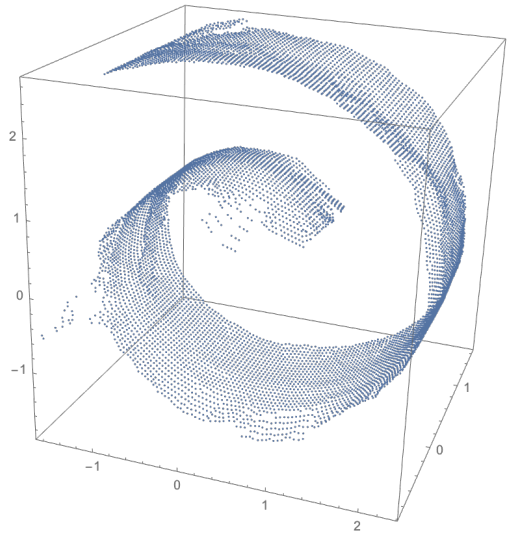
- Overparameterize then tune weight decay
  - Adapts to a broad class of functions
  - Learns a sparse linear combination of learned functions
  - New insight into depth, width and representation

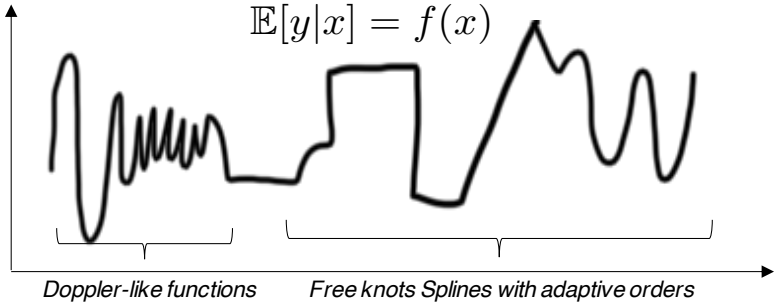
$$n^{-\frac{2\alpha}{2\alpha+d}}$$

- But, does not solve the curse of dimensionality?

Recent work from my group sheds light on how DNNs are adaptive.

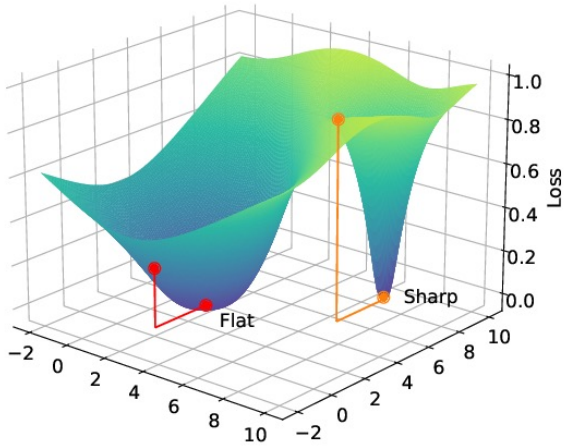
1. “Are Weight Decayed DNNs locally adaptive?”



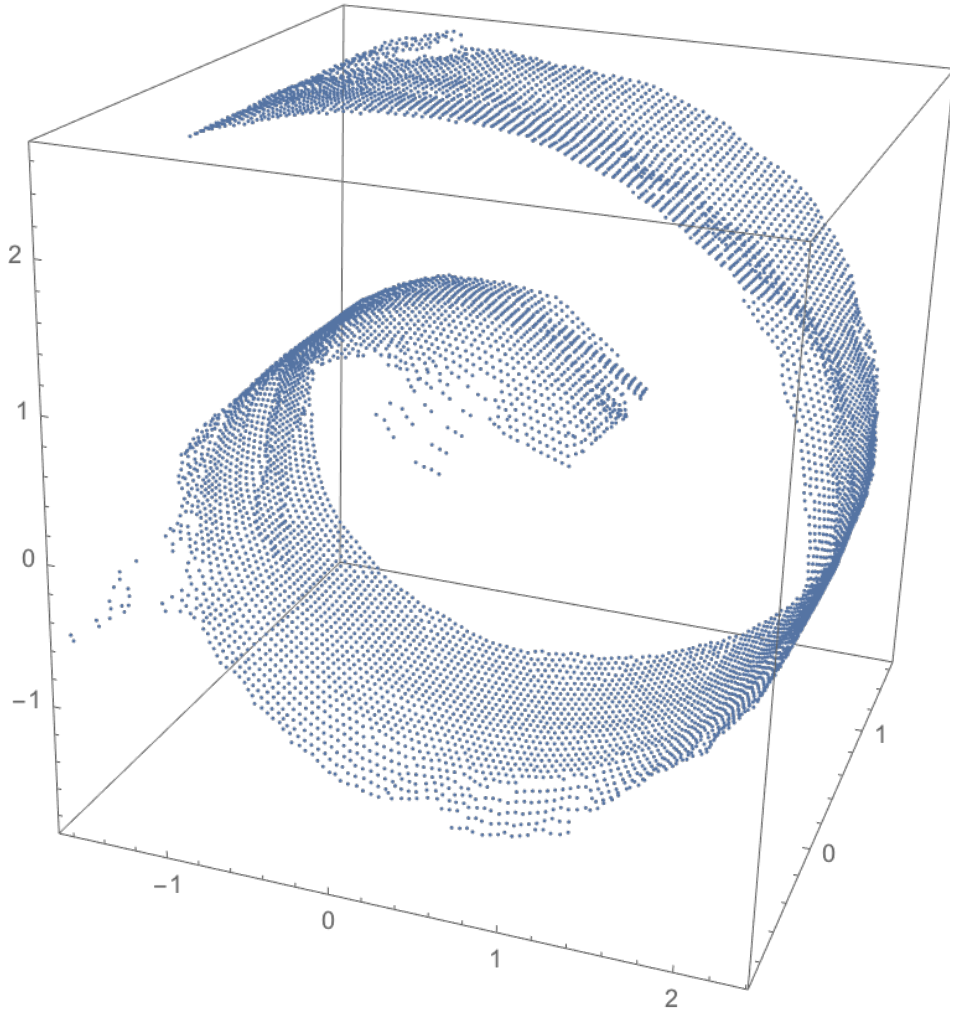


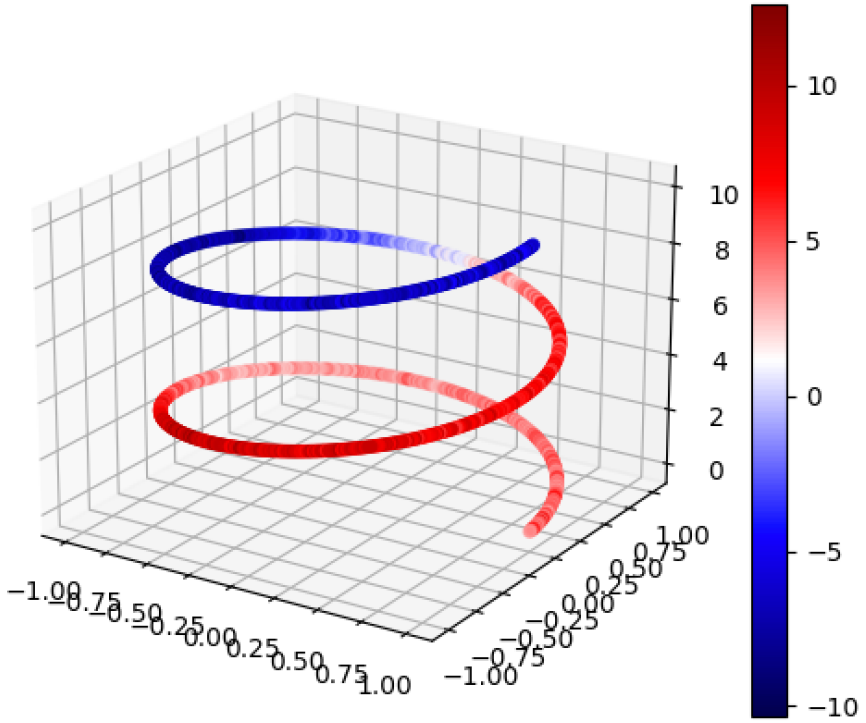
2. “Weight Decayed DNNs adapt to low-dimensional manifolds” (will skip)

3. Generalization by Large Learning Rate in Gradient Descent



Low-dimensional manifolds in high-dimensional space





We prove near-optimal rate that adapts to intrinsic dimension by overparameterization + weight decay

Theorem: ResNeXt, or ConvResNeXt (or PNNs). Lipschitz loss functions, bounded outputs. Choose depth L = log(n), MN = O(n), then:

$$\mathbb{E}_{\mathcal{D}}[\mathrm{Loss}(\hat{f}(\boldsymbol{x}), y)] \leq \mathbb{E}_{\mathcal{D}}[\mathrm{Loss}(f^*)] + \tilde{O}(n^{-\frac{\alpha/d}{2\alpha/d+1}(1-o(1))}),$$

- No (exponential) dependence on the ambient dimension D.
- No need to know intrinsic dim d, and other parameters
- Tune only the weight decay

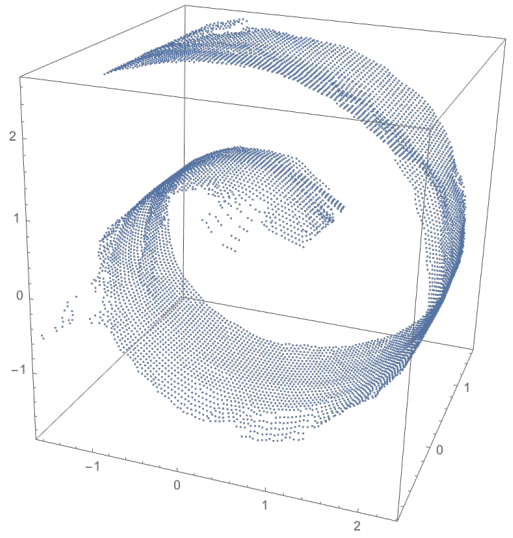
Checkpoint: Parallel NNs adapt to low-dimensional manifolds in data

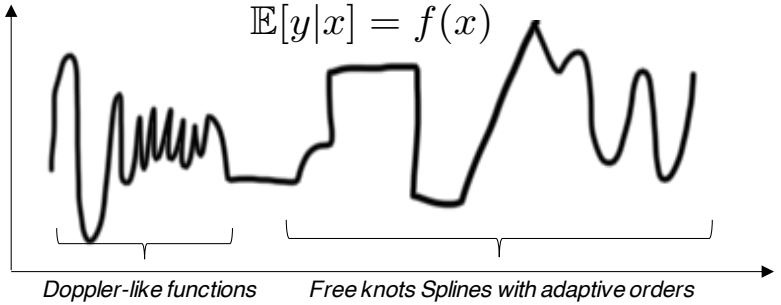
- Overcomes the curse of dimensionality
- Adapts to unknown low-dimensional structures by tuning only weight decay
- ConvNet also works (as long as there are parallel or sequential structures, e.g., ResNet or ResNeXt)

- But, still says nothing about computation…

Recent work from my group sheds light on how DNNs are adaptive.

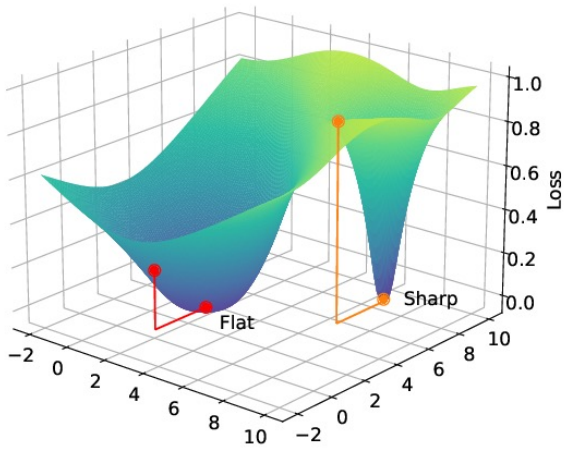
1. “Are Weight Decayed DNNs locally adaptive?”



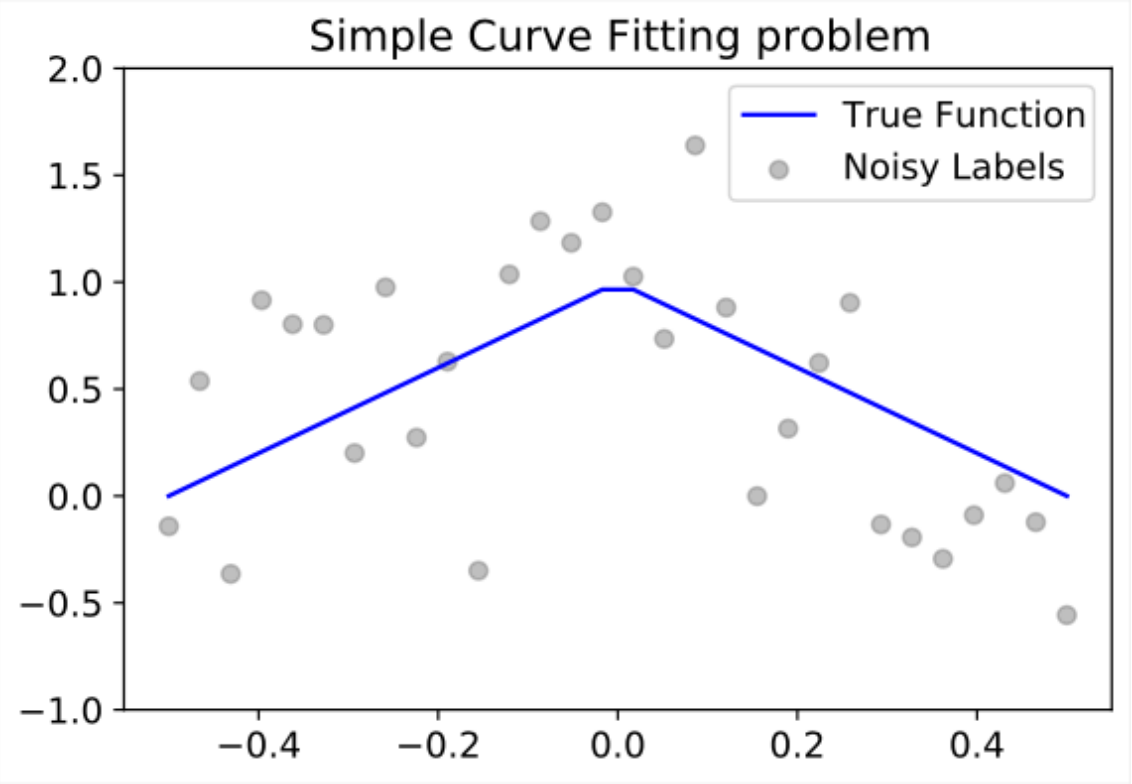


2. “Weight Decayed DNNs adapt to low-dimensional manifolds” (will skip)

3. Generalization by Large Learning Rate in Gradient Descent



Let's start with an example



30 data points. Noisy labels.
2-Layer ReLU NN with 1000 neurons.
Minimizing square loss.
No regularization.

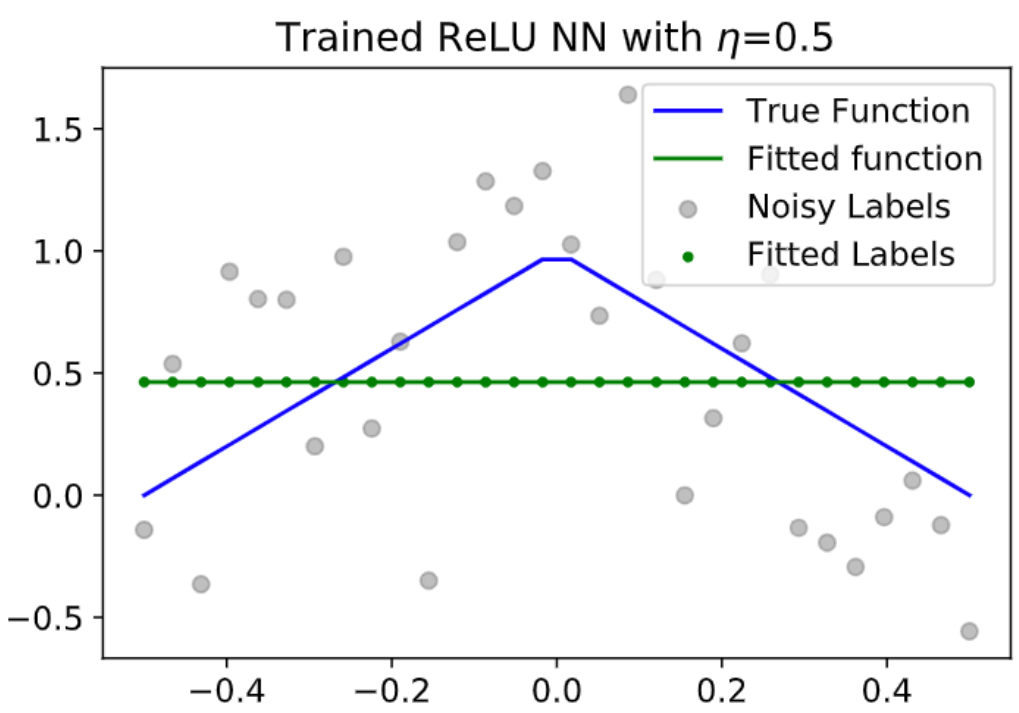
Which ones of the following are true?

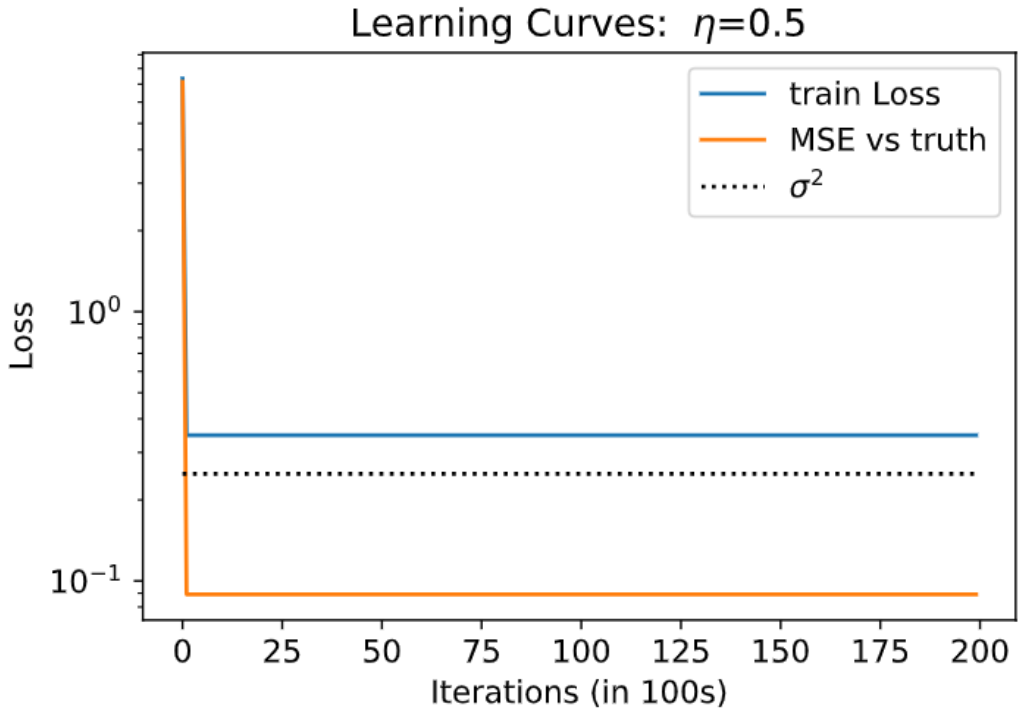
A. Global optimal solution has 0-loss.

B. Gradient descent finds global optimal ("interpolating") solutions.

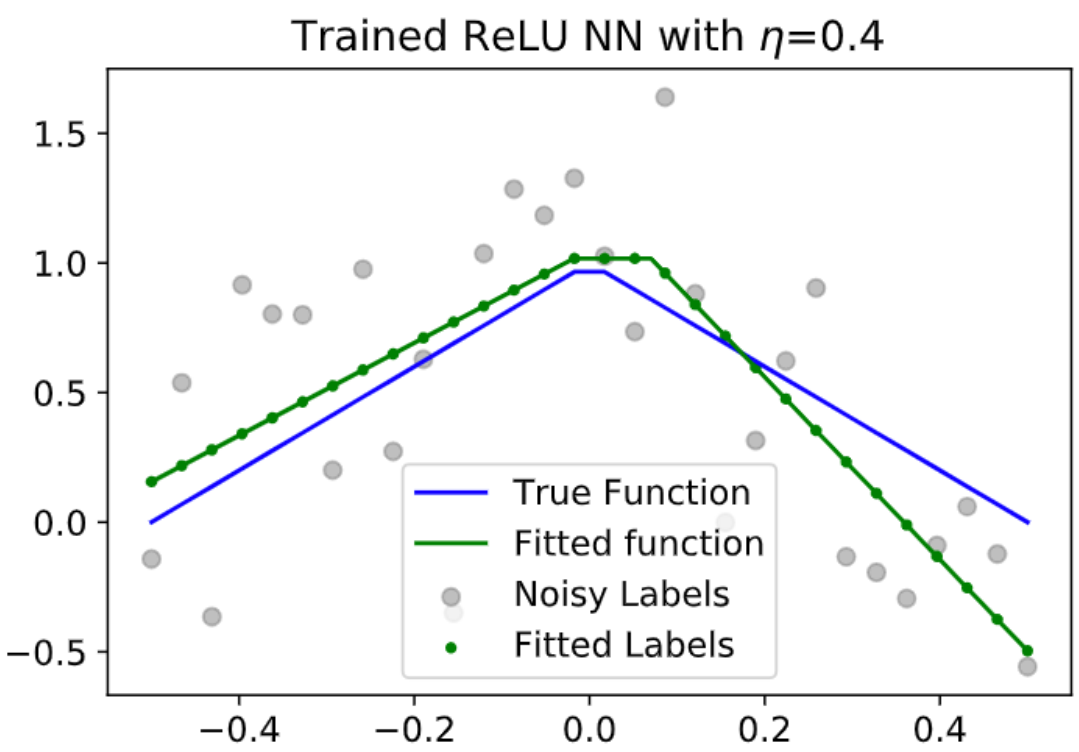
C. GD solution satisfies "Benign overfitting"

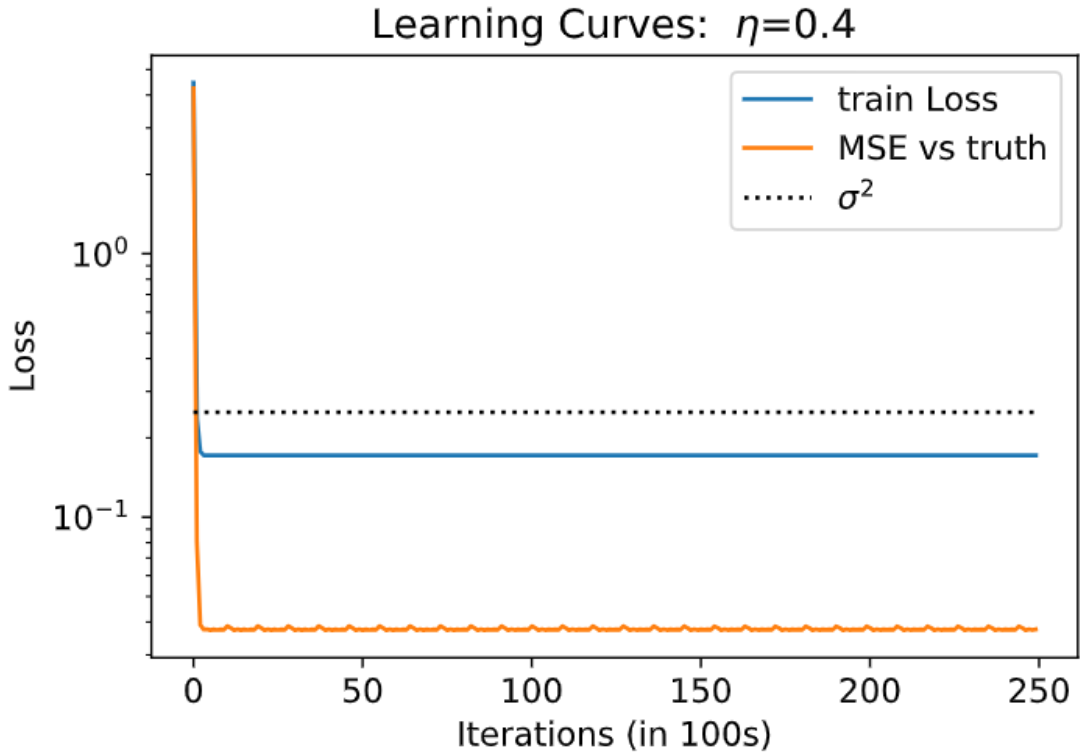
It’s surprisingly difficult to find an interpolating solution with gradient descent.



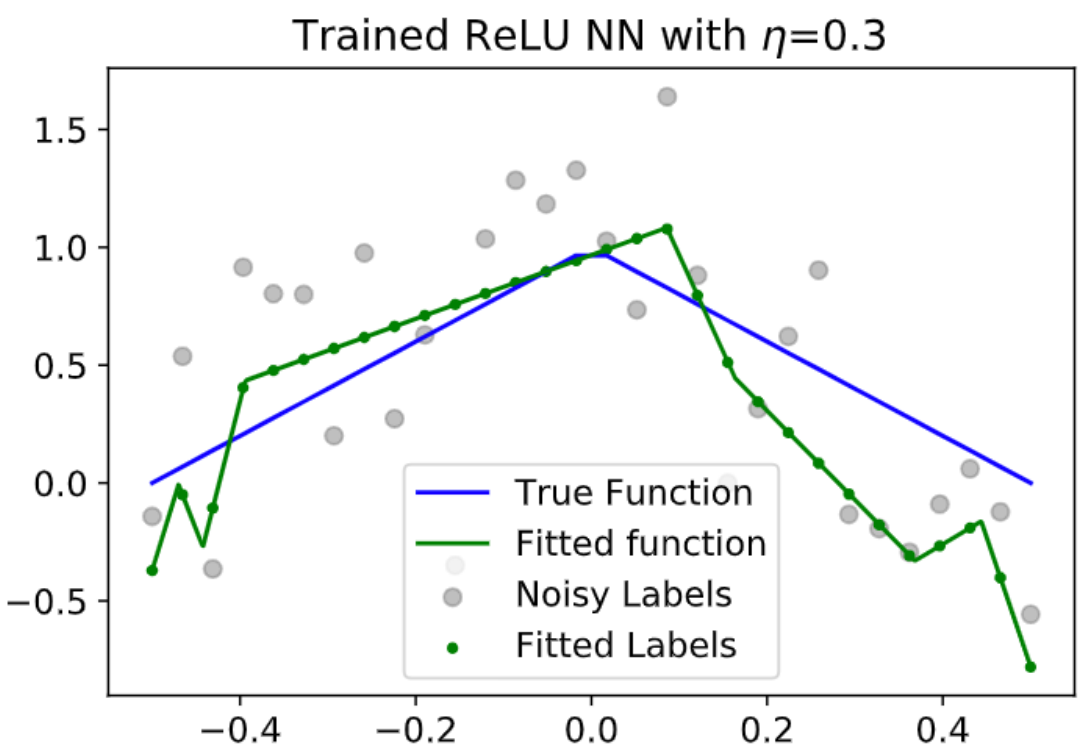


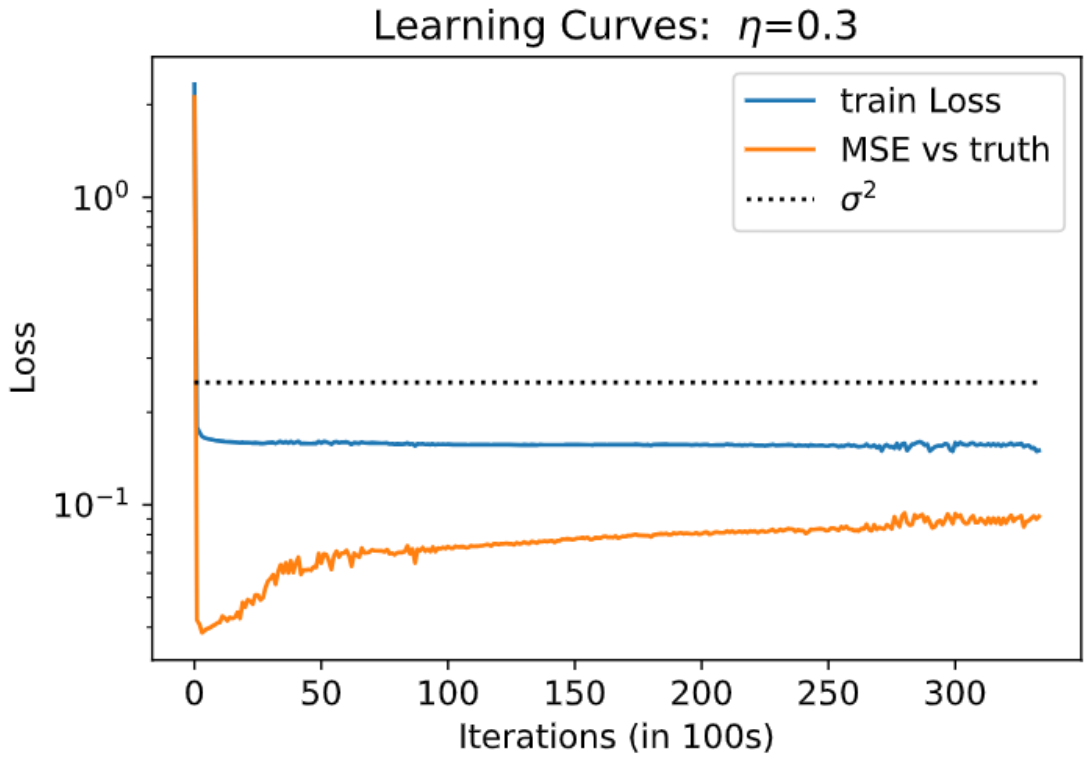
Learning Rate = 0.4



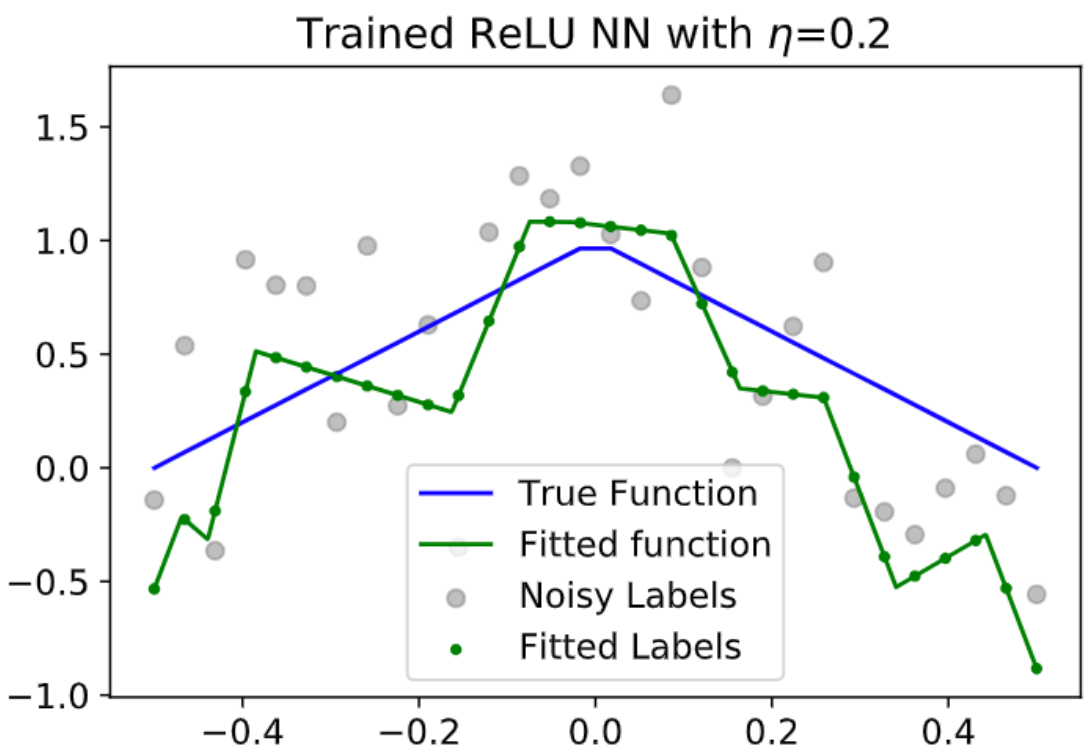


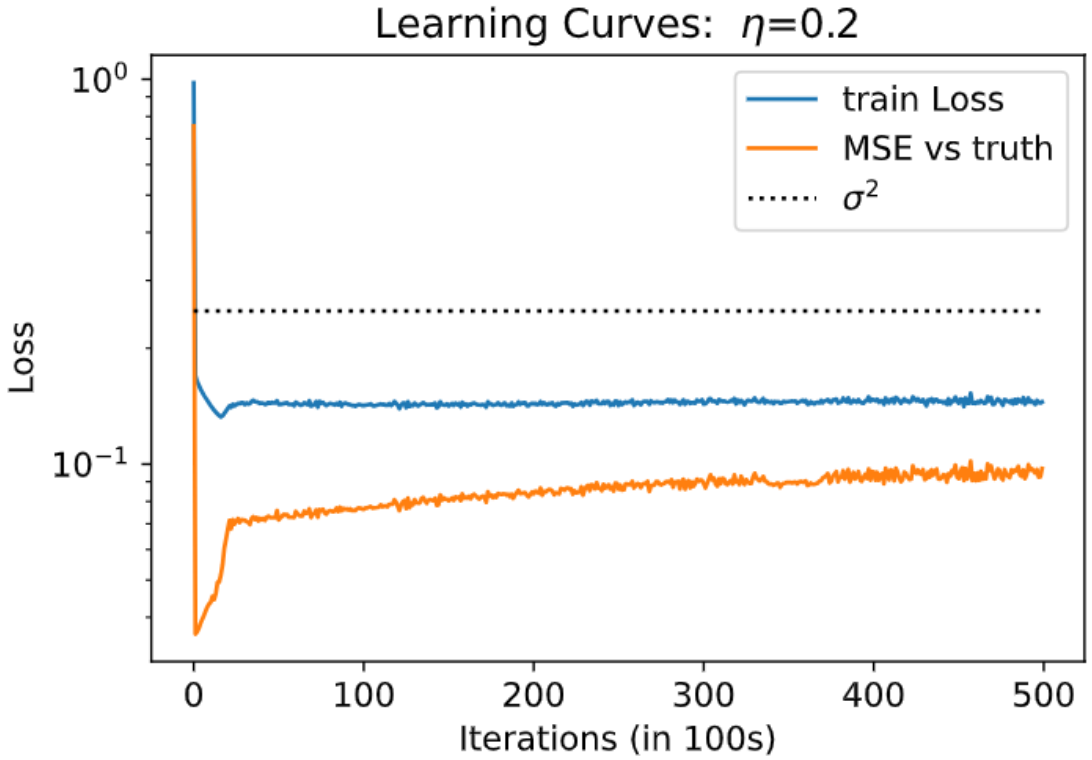
Learning Rate = 0.3



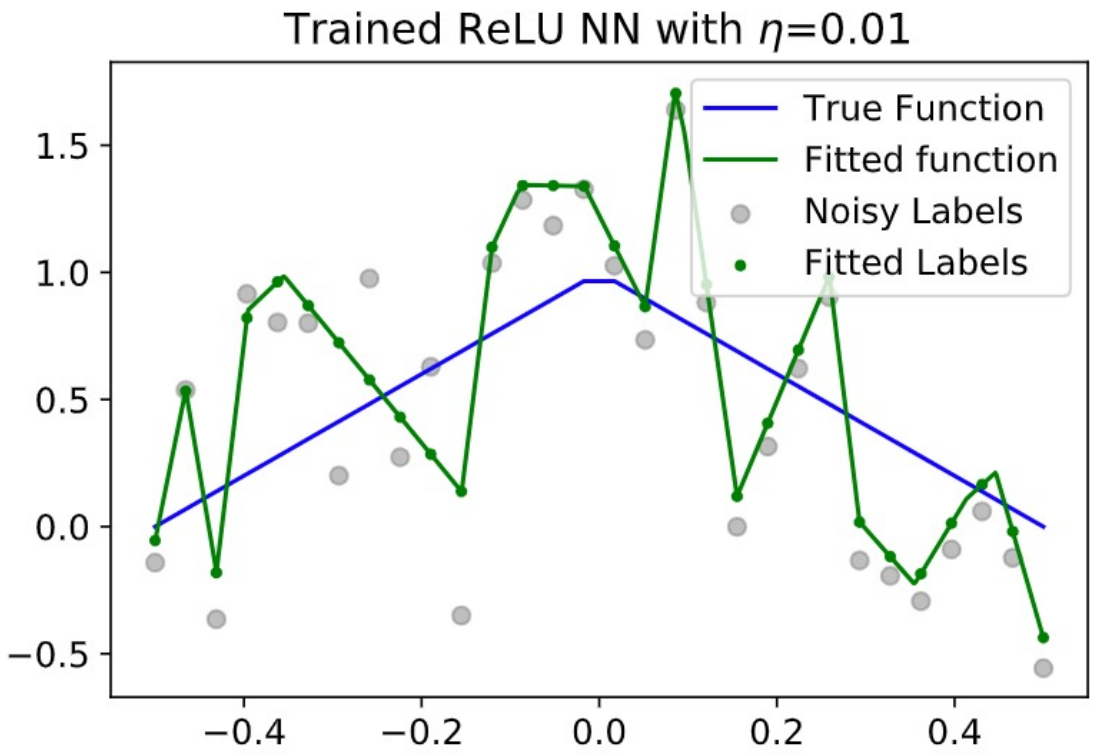


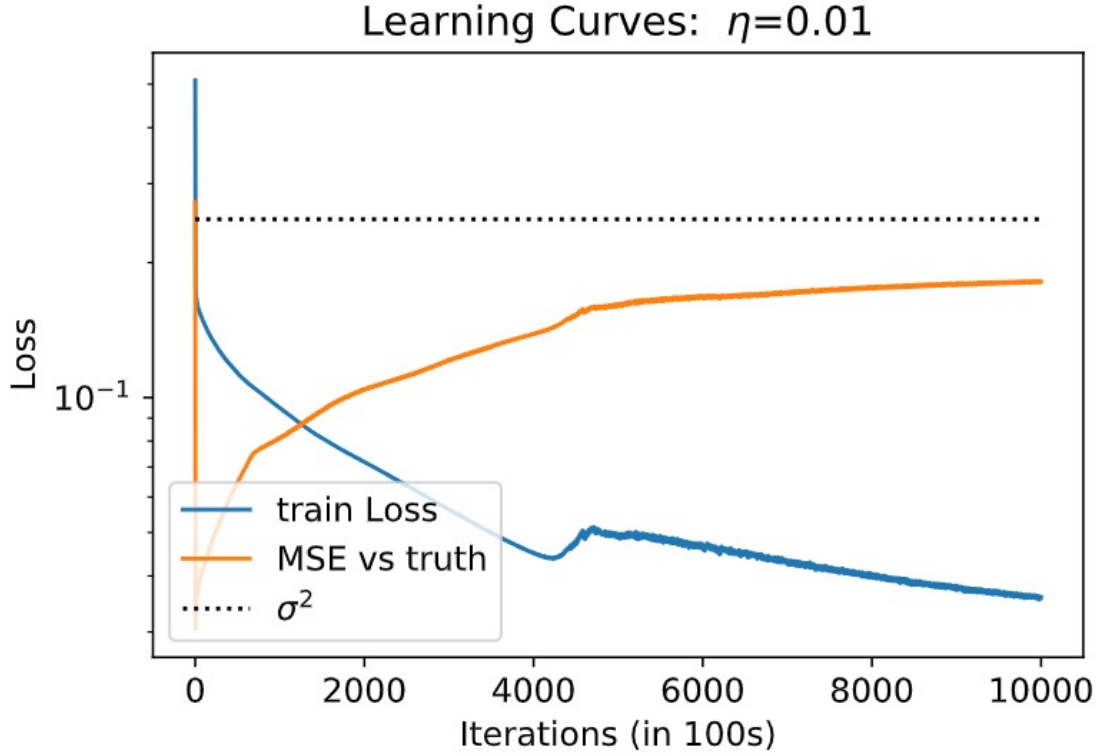
Learning Rate = 0.2



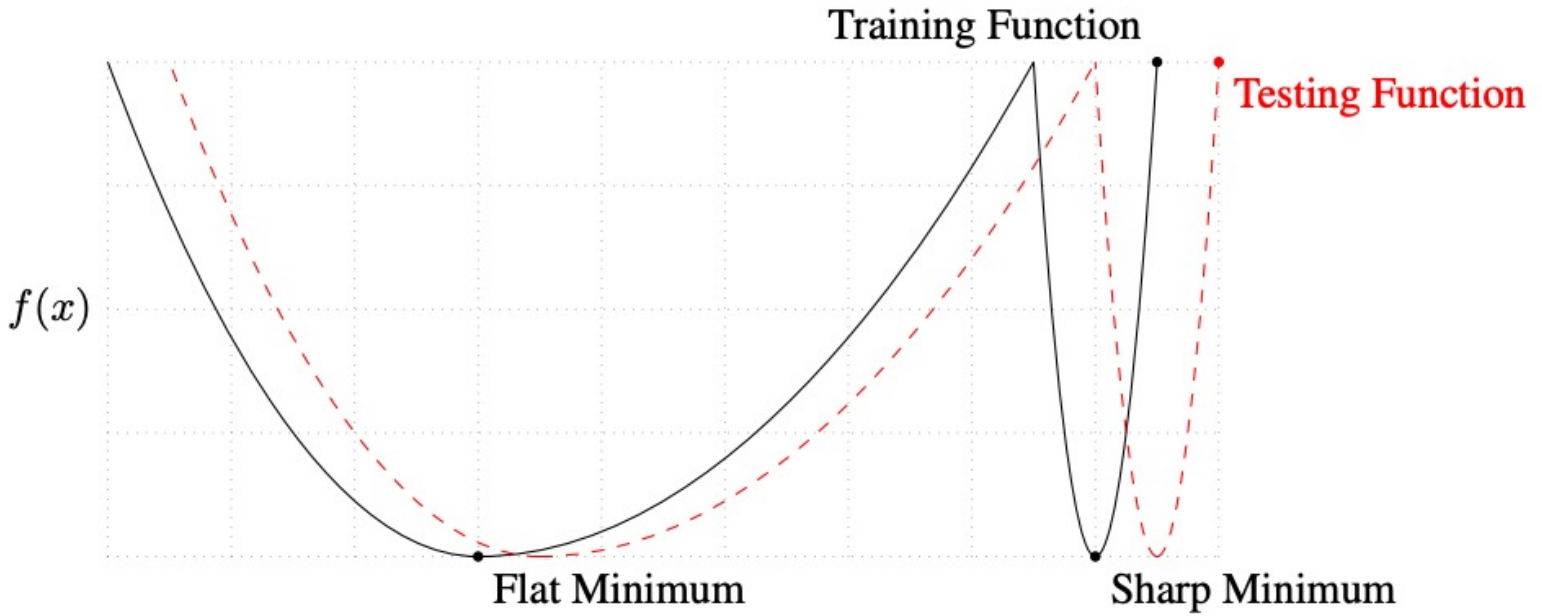


Learning Rate = 0.01





Minima stability of Gradient Descent and SGD

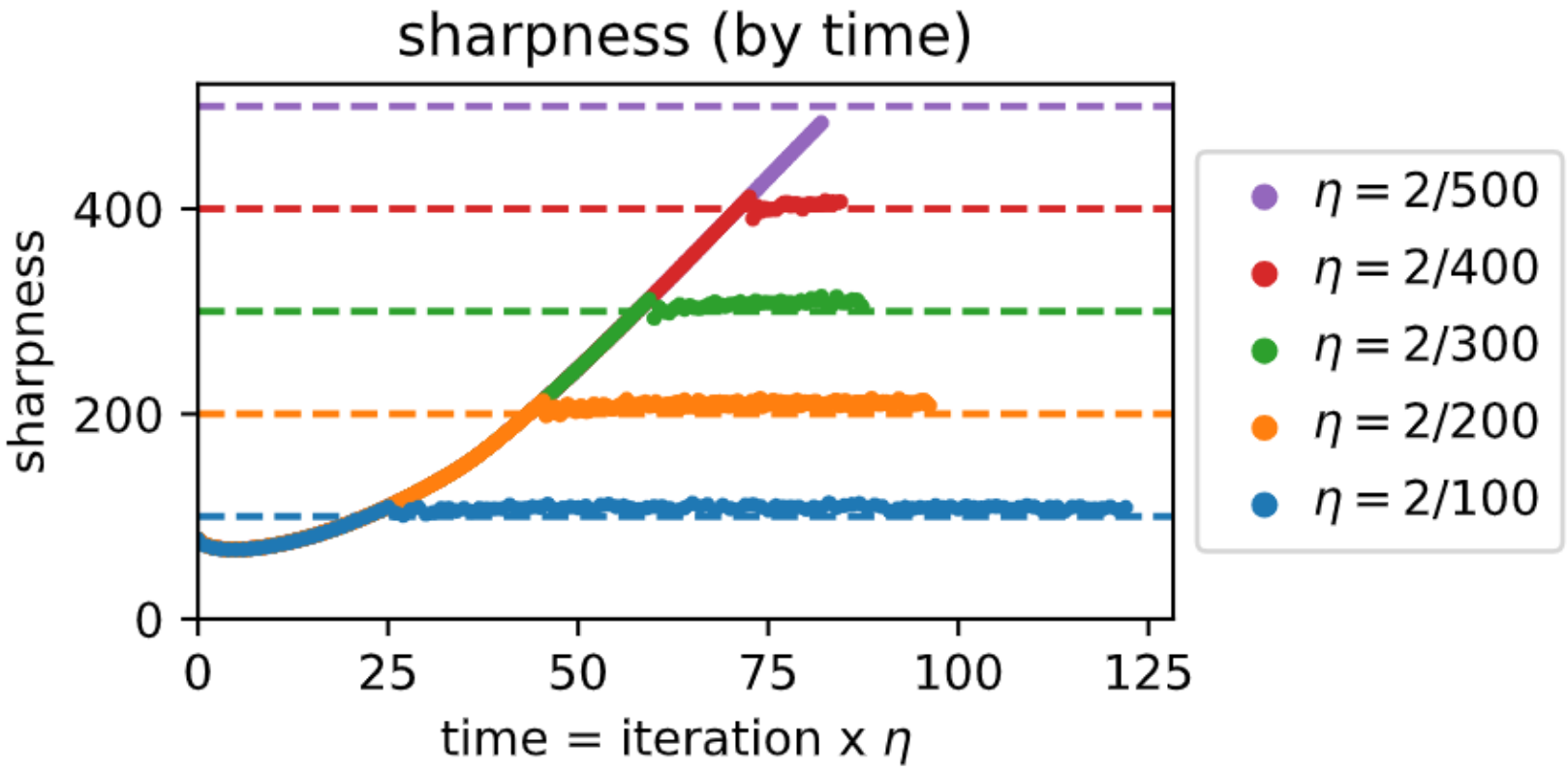


$$x_{t+1} = x_t - \eta \nabla Loss(x_t)$$

- Need learning rate $\eta \leq 2/\beta$ to converge for ***β*-smooth** functions. **Stable minima** are those that GD can converge to.

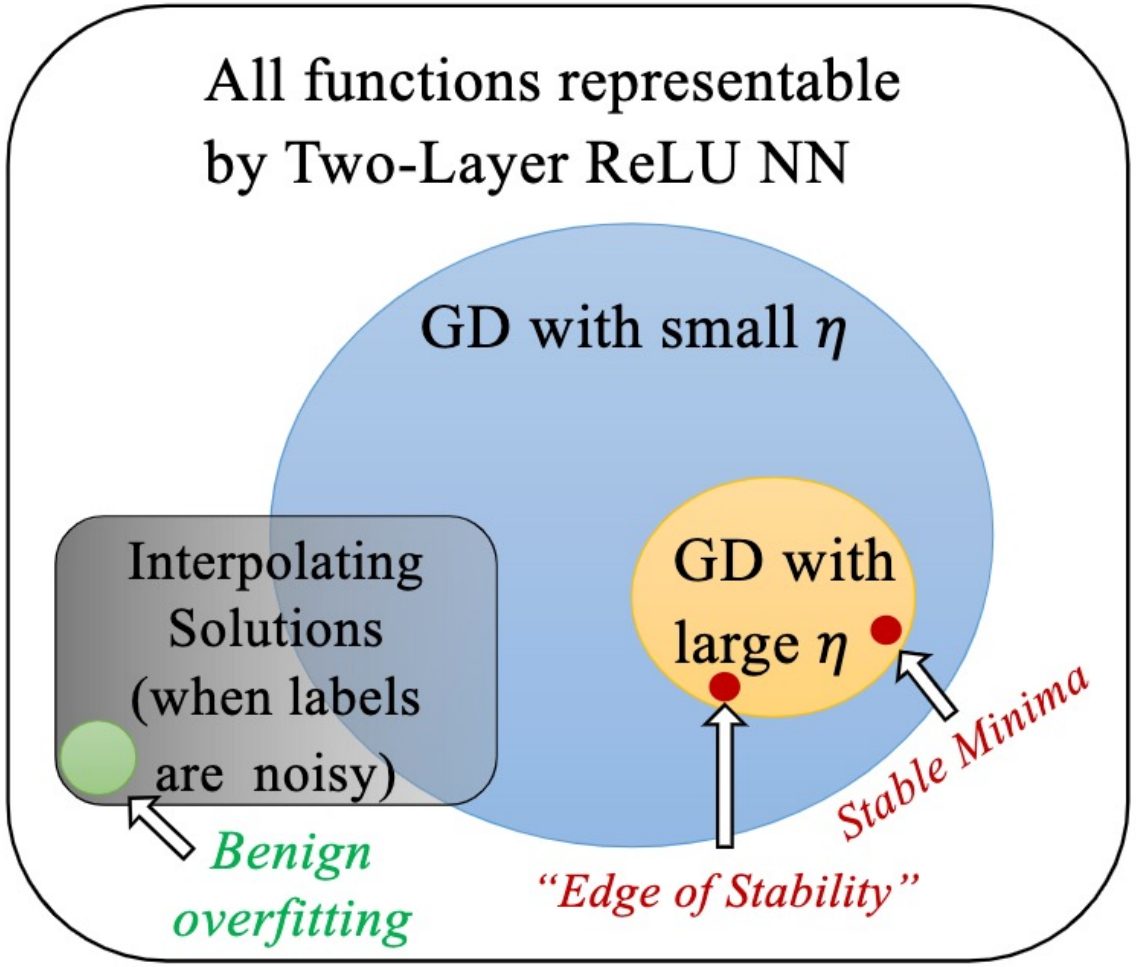
(Wu et al. 2018, Mulayoff et al. 2021)

“Edge of Stability” regime



Cohen et al (2021) Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability

Gradient Descent selects a set of points it can converge to (or stabilize around).



What does this set of functions look like?
A Weighted TV1 class.

$$\left\{ f_\theta \middle| \lambda_{\max}(\nabla^2 \mathcal{L}(\theta)) \leq 2/\eta \right\}$$

$$\subseteq$$

$$\left\{ f \middle| \int |f''(x)| g(x) dx \leq \frac{2}{\eta} \right\} =: \mathrm{TV}_g^{(1)}(C)$$

where $C = 1/\eta + \tilde{O}(1)$

Mulayoff, Rotem, Tomer Michaeli, and Daniel Soudry. "The implicit bias of minima stability: A view from function space." *NeurIPS'2021*

Flatness (in parameter space) implies smoothness (in function space)

Theorem (Qiao, Zhang, Singh, Soudry and W., 2024): Consider a ReLU activated two-layer NN / square loss. Let f be any function represented by the neural network parameterized by θ. Assume f_θ is twice differentiable at θ. ~~Assume f interpolates~~.

$$\int_{-x_{\max}}^{x_{\max}} |f''(x)|g(x)dx \leq \frac{\lambda_{\max}(\nabla_\theta^2 \mathcal{L}(\theta))}{2} - \frac{1}{2} + x_{\max}\sqrt{2\mathcal{L}(\theta)},$$

Assume data is coming from $\boldsymbol{y_i = f_0(x_i) + noise}$, then w.h.p.

$$\int_{-x_{\max}}^{x_{\max}} |f''(x)|g(x)dx \leq \frac{\lambda_{\max}(\nabla_\theta^2 \mathcal{L}(\theta))}{2} - \frac{1}{2} + \widetilde{O}\left(\sigma x_{\max} \cdot \min\left\{1, \sqrt{\frac{k}{n}}\right\}\right) + x_{\max}\sqrt{\mathrm{MSE}(f)}.$$

- Tune learning rate => select smoothness of f
- Smoothness of f => Generalization bounds

Key proof idea: Hessian Decomposition

Interpolation helps here

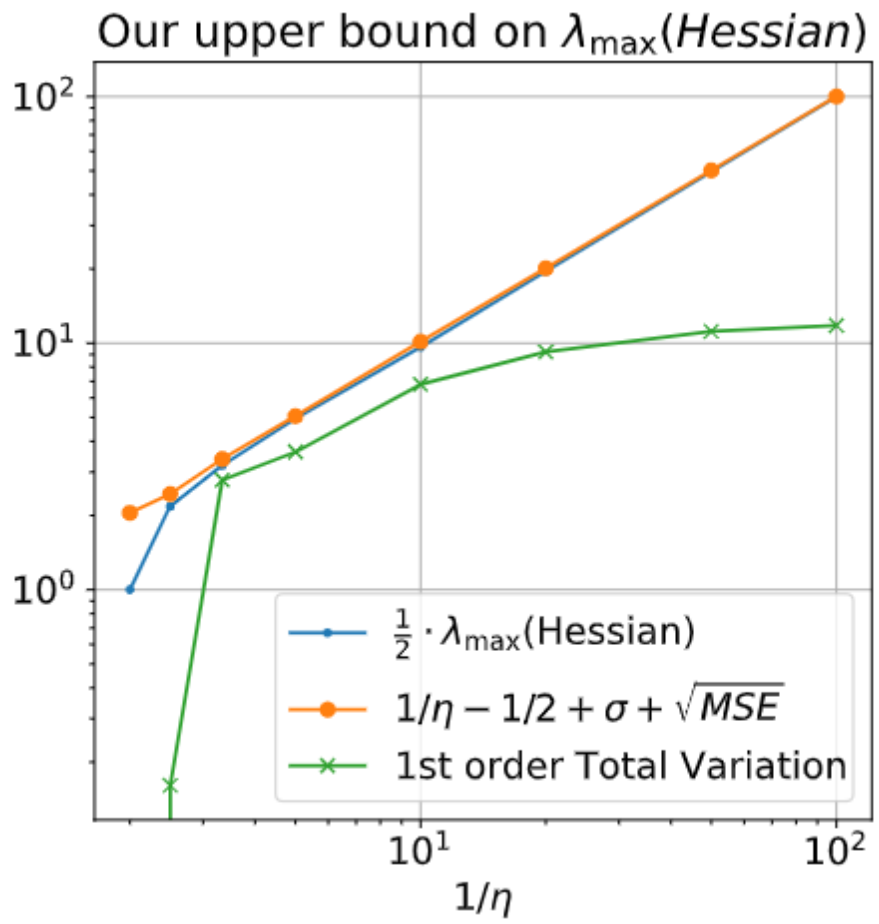
$$\nabla_\theta^2 (f_\theta(x) - y)^2 = \nabla_\theta f_\theta(x) \nabla_\theta f_\theta(x)^T + (f_\theta(x) - y) \nabla_\theta^2 f_\theta(x)$$

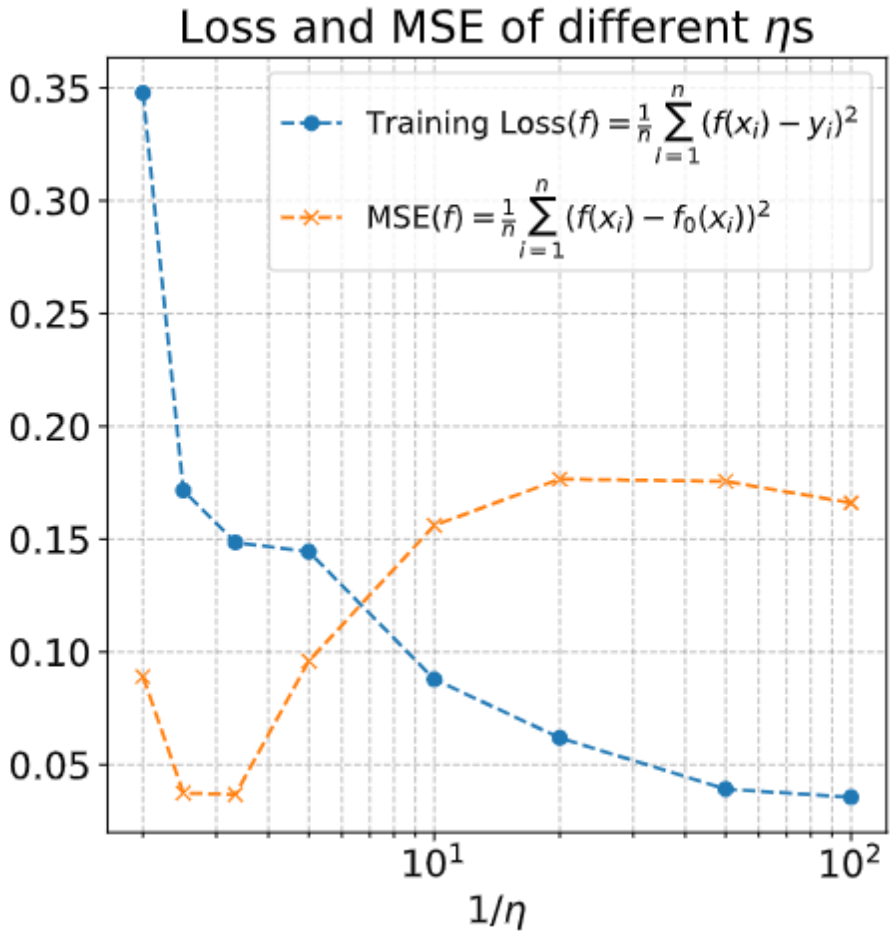
(Mulayoff et al. 2021) showed that:

$$\lambda_{\max}\left(\frac{1}{n}\sum_i \nabla_\theta f_\theta(x_i) \nabla_\theta f_\theta(x_i)^T\right)$$

is connected to the TV1 of f_θ.

1/η very precisely predicts the sharpness, and gives a classical U-shape risk curve.





Generalization bounds that stem from these function space characterization

Theorem: We proved that in the **strict interior of the data support:**

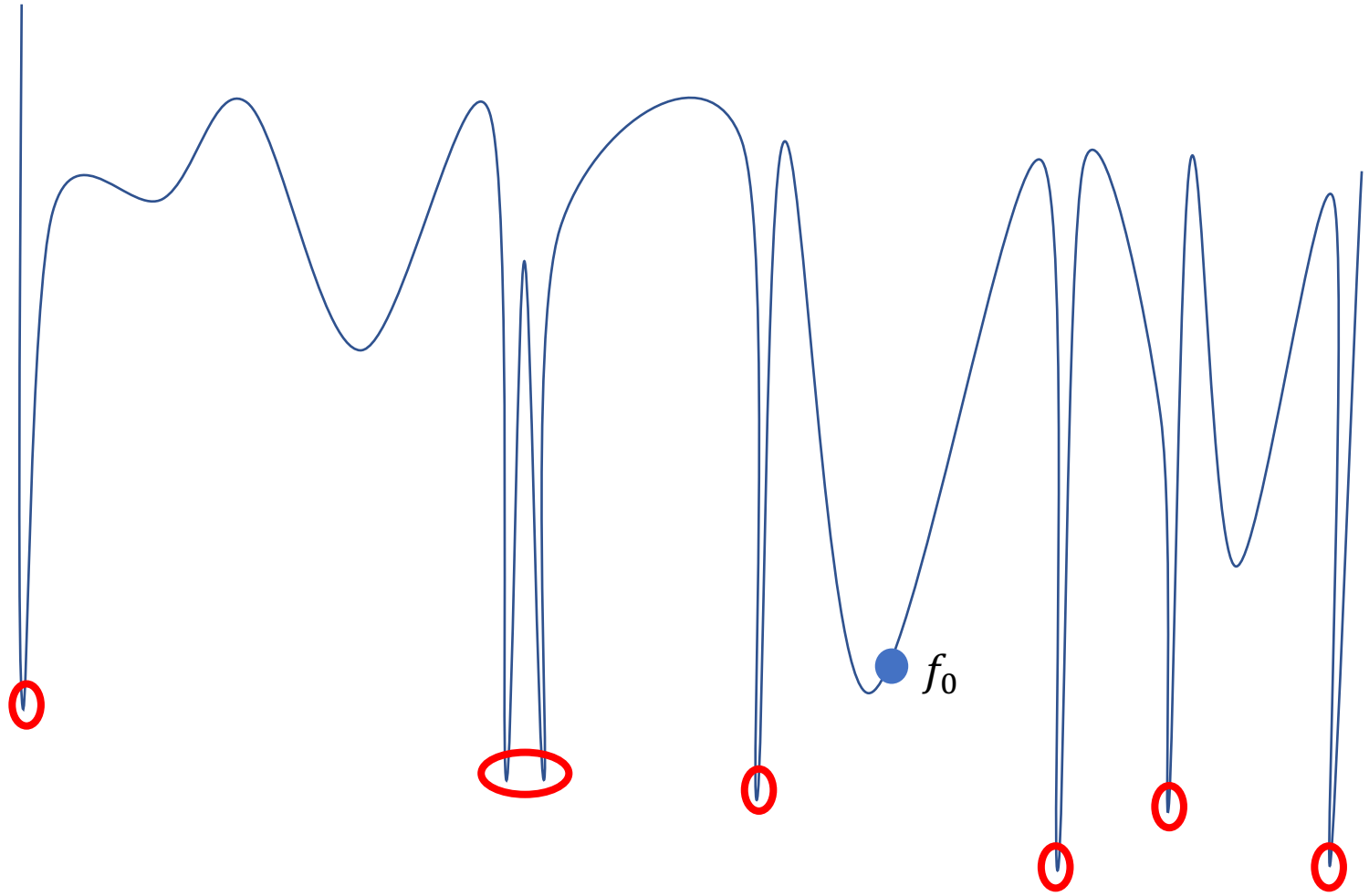
1. Agnostic case, generalization gap = O(n^{-2/5})

2. If training loss smaller than σ^2 then w.h.p., get an MSE

$$\mathrm{MSE}_{\mathcal{I}}(f) = \frac{1}{n_{\mathcal{I}}} \sum_{x_i \in \mathcal{I}} (f(x_i) - f_0(x_i))^2 \leq \widetilde{O}\left(\left(\frac{\sigma^2}{n_{\mathcal{I}}} \right)^{\frac{4}{5}} \left(\frac{x_{\max}}{\eta} + \sigma x_{\max}^2 \right)^{\frac{2}{5}} \right)$$

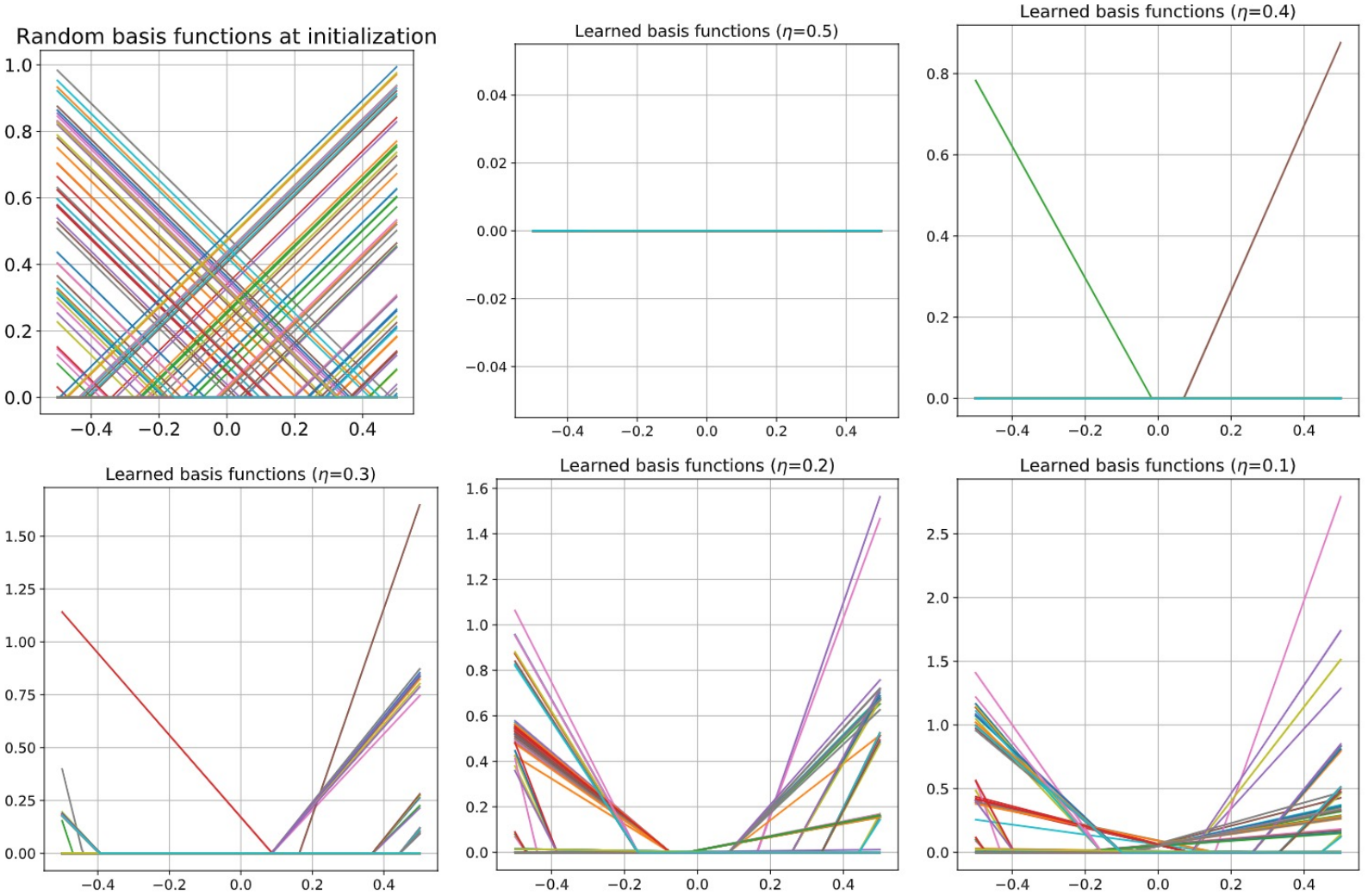
*Again, near minimax optimal, beating kernels.

It tells us something new about the energy landscape of overparameterized NN training on noisy problems



Training with GD automatically avoids these sharp solutions

“Representation learning” becomes learning knots, and knot “sparsity” but are they really zero?



Take-home messages

- In simple “curve fitting” problem, appropriately tuned DNN models can be locally adaptive
  - Not revealed by NTK theory
  - Not compatible with Benign overfitting
- Adaptivity advantage
  - Tuning weight decay / Learning rate
- Implicit sparsity in a learned dictionary space
  - Analysis vs Synthesis

Implications on modern generalization theory?

Classical generalization theory

VC-theory
Rademacher complexity
Metric entropy

Other empirical process theory for getting Uniform convergence

limitations
- Nothing about computation
- Unform convergence might not be needed

In between

Algorithmic stability

Minima stability

Edge-of-Stability

…

Modern generalization theory

NTK theory
Benign overfitting

Everything to do with optimization.

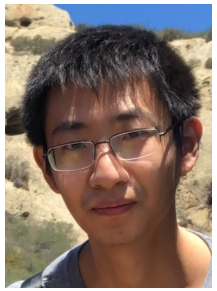
limitations
Often restrictive in model classes / data distribution
Dynamic theory is hard for complex architectures.

Thank you for your attention!

References:

- Zhang et al. (2022) **Deep Learning meets Nonparametric Regression: Are Weight-Decayed DNNs Locally Adaptive?** https://arxiv.org/abs/2204.09664
- Zhang et al. (2023) **Nonparametric Classification on Low Dimensional Manifolds using Overparameterized Convolutional Residual Networks:** https://arxiv.org/abs/2307.01649
- Qiao et al. (2024) **Stable Minima Cannot Overfit in Univariate ReLU Networks: Generalization by Large Step Sizes:** https://arxiv.org/abs/2406.06838

- Work partially supported by
  - NSF DMS Grant "SCALE MoDL: The Adaptivity of Deep Learning"







Kaiqi Zhang, Dan Qiao, Esha Singh, Zixuan Zhang, Minshuo Chen, Tuo Zhao and Daniel Soudry