Deep Learning meets Nonparametric Regression: Are *Weight Decayed* DNNs locally adaptive?

Yu-Xiang Wang

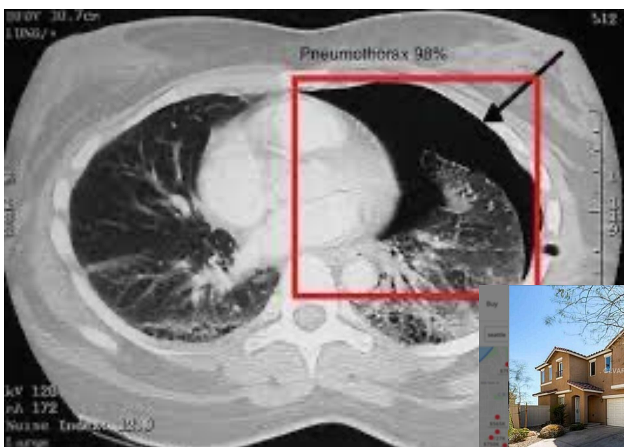
Joint work with Kaiqi Zhang →

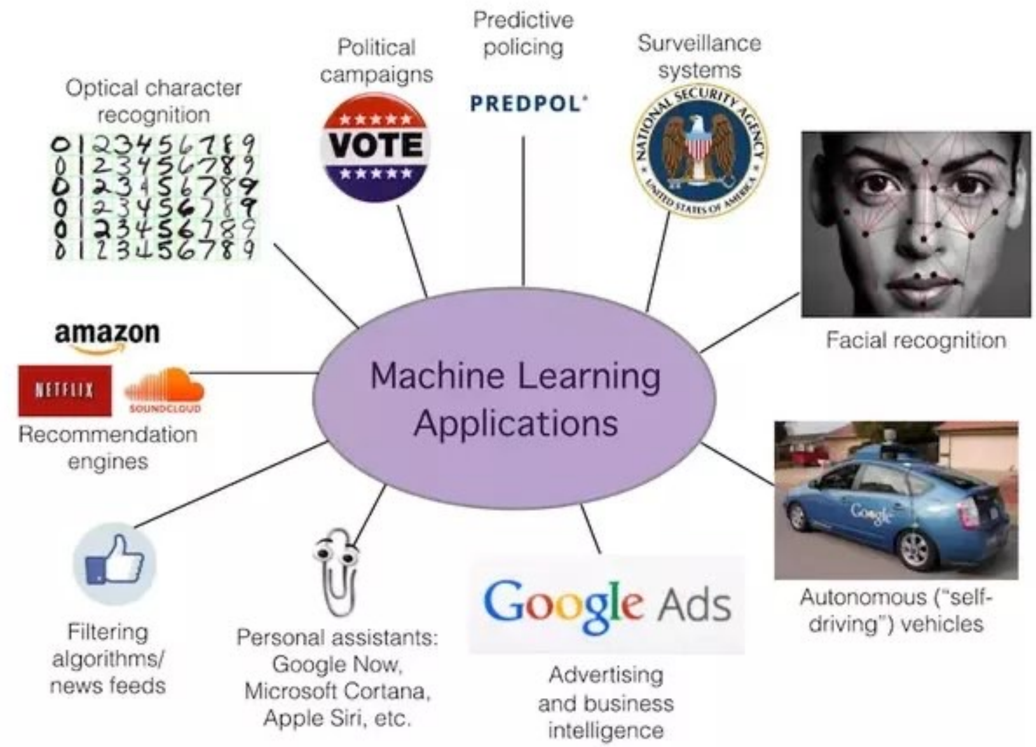
COMPUTER SCIENCE
UC SANTA BARBARA
Computing. ReInvented.

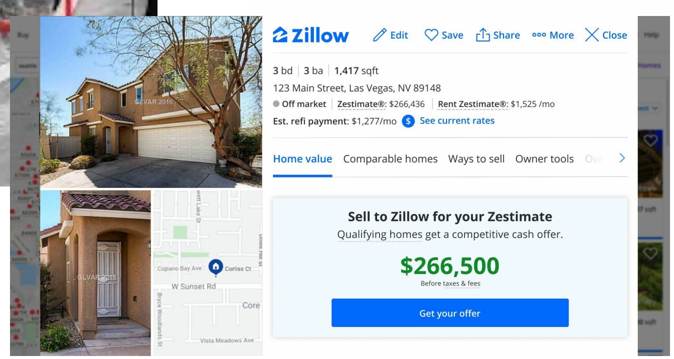
Outline

- Motivation
  - Mysteries around deep neural networks
  - Probe it from nonparametric regression angle
- Warm-up
  - 2-layer NN with weight decay vs LAR Splines
- Main results
  - L-layer parallel NN vs Sparse Linear Regression
  - Error bound and discussion
- Proof sketch

~~AI~~ Machine Learning has revolutionized almost every aspect of our daily life







Deep Neural Networks (DNN) is the main workhorse behind many breakthroughs.

Feedforward Neural Net (FFN)

- also known as multilayer perceptron (MLP)

$x \in \mathbb{R}^d$

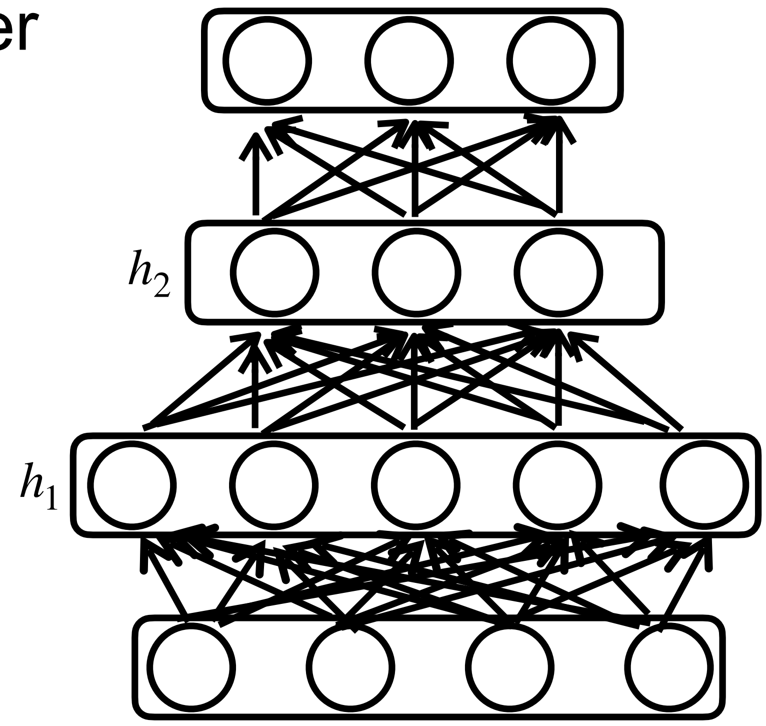
$h_1 = \sigma(w_1 \cdot x + b_1) \in \mathbb{R}^{d_1}$

$h_l = \sigma(w_l \cdot h_{l-1} + b_l) \in \mathbb{R}^{d_l}$

$o = \text{Softmax}(w_L \cdot h_{L-1} + b_L)$

Parameters

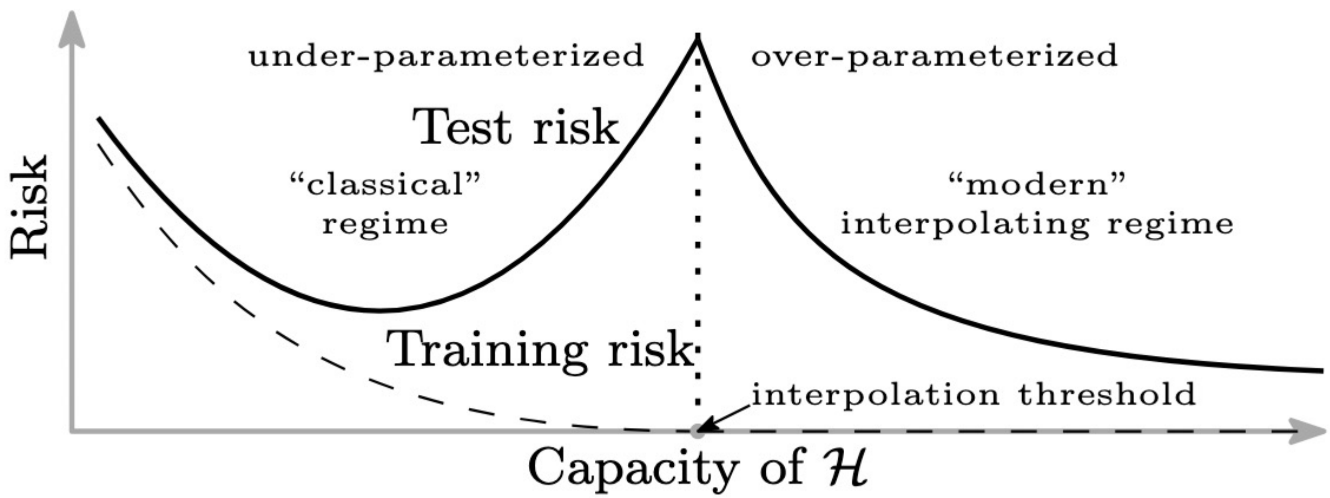
$\theta = \{w_1, b_1, w_2, b_2, \ldots\}$



Slide / illustration taken from Lecture 5 of my course with Lei Li.

From the statistical point of view, the success of DNN is a mystery.

- Observe that:
  - Way more parameters than you have data to fit them
  - *Appears* to not follow classical Bias-Variance tradeoff



(Figure from Belkin et al. (2018) "Double Descent")

  - Highly nonconvex, yet optimization seems to be easy with SGD

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 (e.g., Bartlett et al. 2020)

The “adaptivity” conjecture

- 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

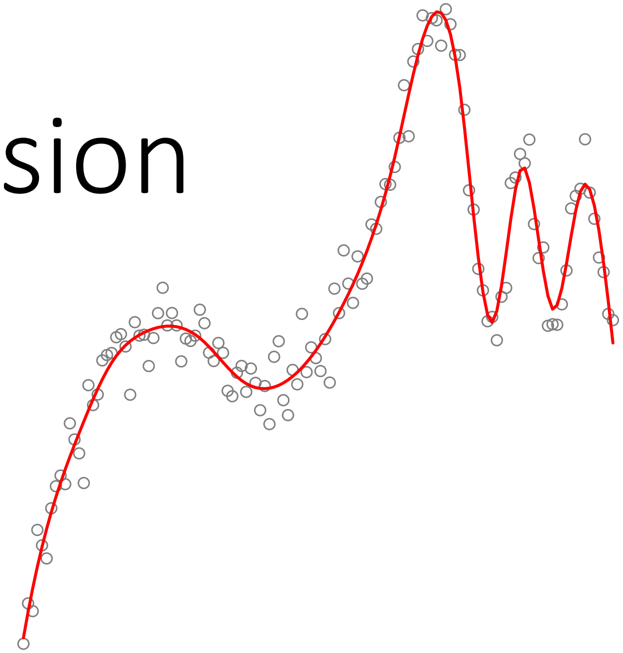
Nonparametric regression

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

Goal: Estimate the function using noisy data

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

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



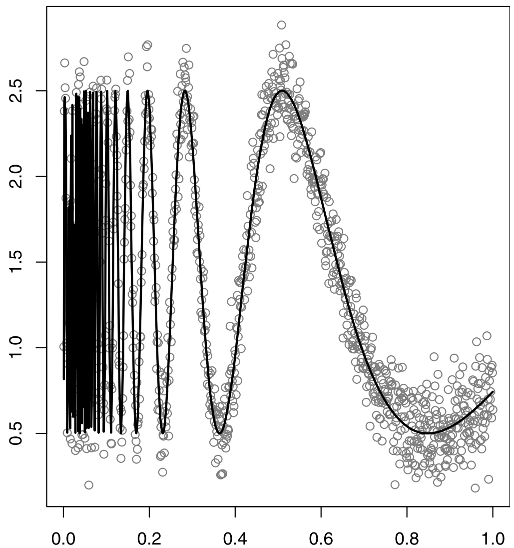
- 50+ years of associated literature

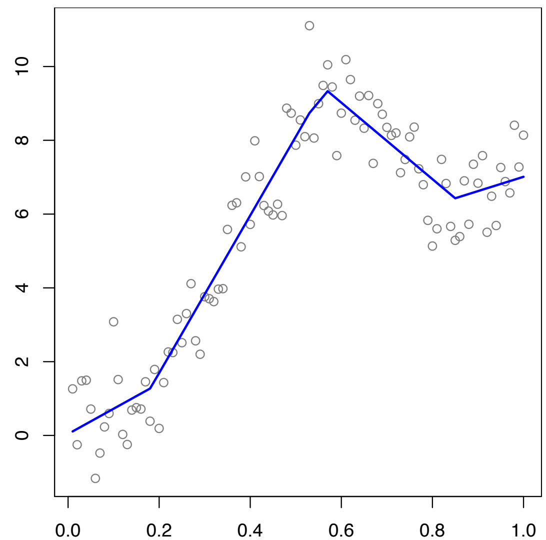
  [Nadaraya, Watson, 1964]

  - Kernels, splines, local polynomials
  - Gaussian processes and RKHS
  - CART, neural networks

- Also known as smoothing, signal denoising /filtering in signal processing & control.

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.

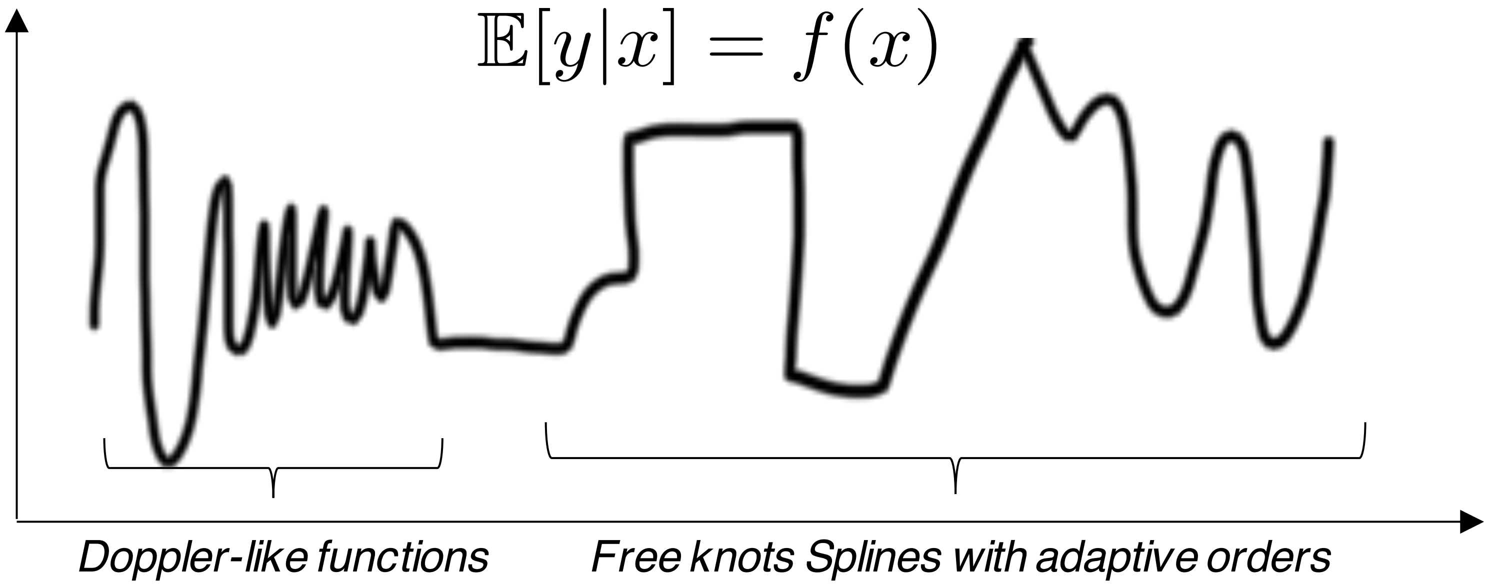
NTK are strictly suboptimal for locally adaptive nonparametric regression

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

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

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

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: 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.*

Outline

- Motivation
  - Mysteries around deep neural networks
  - Probe it from nonparametric regression angle

- Warm-up
  - 2-layer NN with weight decay vs LAR Splines

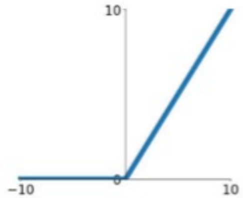
- Main results
  - L-layer parallel NN vs Sparse Linear Regression
  - Error bound and discussion

- Proof sketch

Background: DNN with “ReLU” activations and Weight Decays

- ReLU (Rectified Linear Unit activation)

ReLU
$\max(0, x)$



- “Weight decay” == L2 Regularization

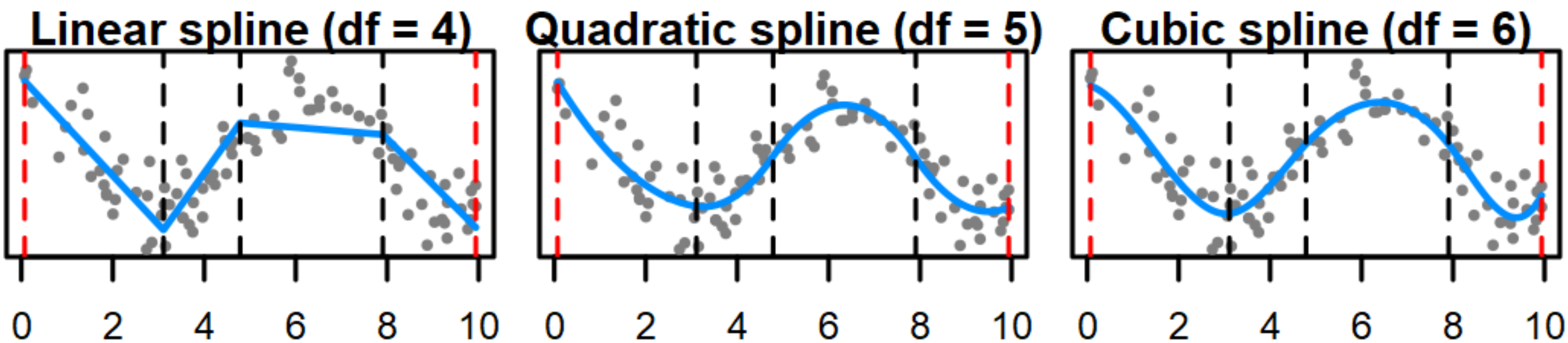
$$\nabla_\theta \left(\mathcal{L}(\theta) + \frac{\lambda}{2} \|\theta\|^2 \right) = \nabla \mathcal{L}(\theta) + \lambda\theta$$

- Gradient Descent:

“Weight decay”

$$\theta_{t+1} = \theta_t - \eta(\nabla \mathcal{L}(\theta_t) + \lambda\theta_t) = \boxed{(1 - \eta\lambda)}\theta_t - \eta\nabla\mathcal{L}(\theta_t)$$

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.
  - **Free-knot splines:** fix the number of knots, but optimize over where to put them.

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

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

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)

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

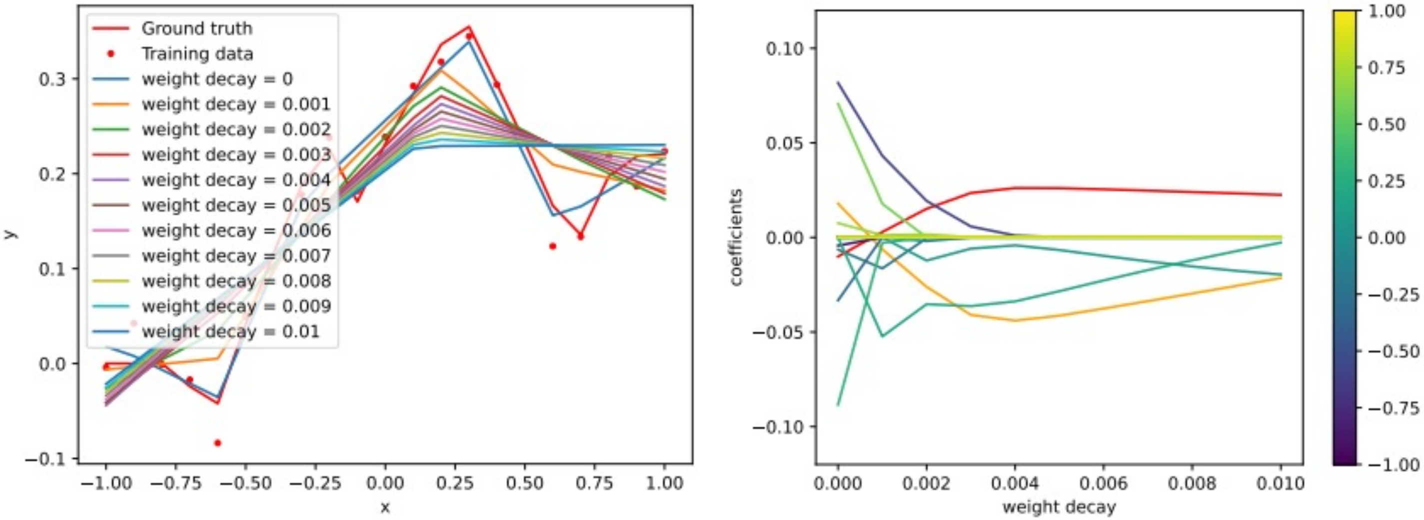
over all functions!

- By Mammen and Van De Geer (1997)

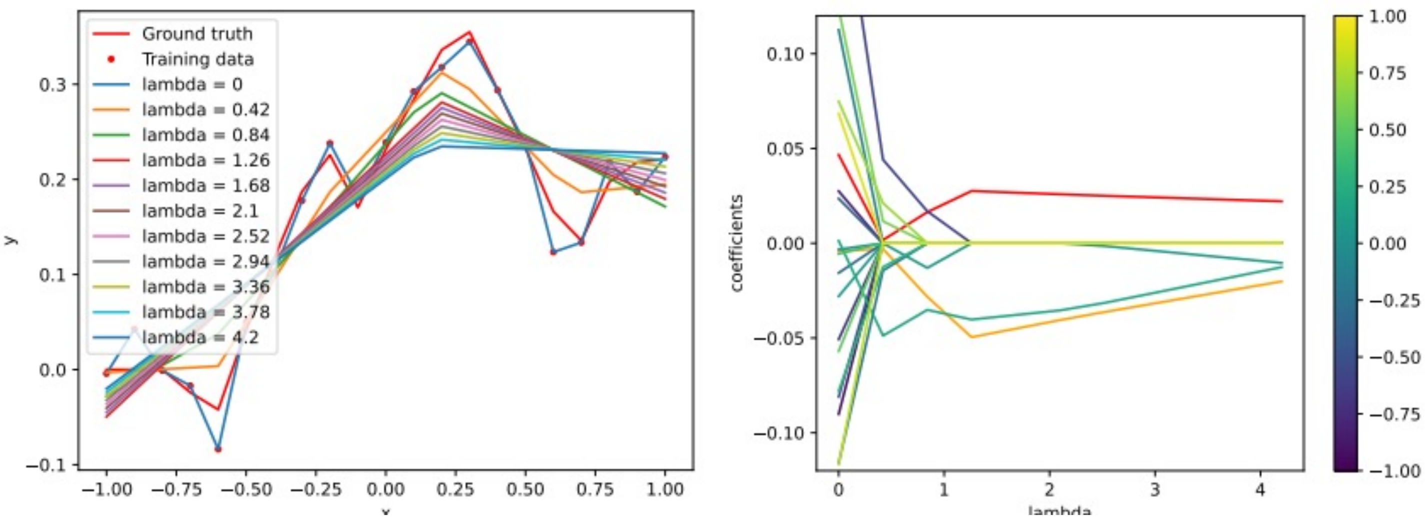
$$\mathrm{MSE}(\hat{f}) = O(n^{-(2m+2)(2m+3)}).$$

The equivalence is also valid empirically.

Weight decayed ReLU NN



L1-trend filtering



(Example for 2 layer ReLU NN + weight decay from Fig 6 of our paper)

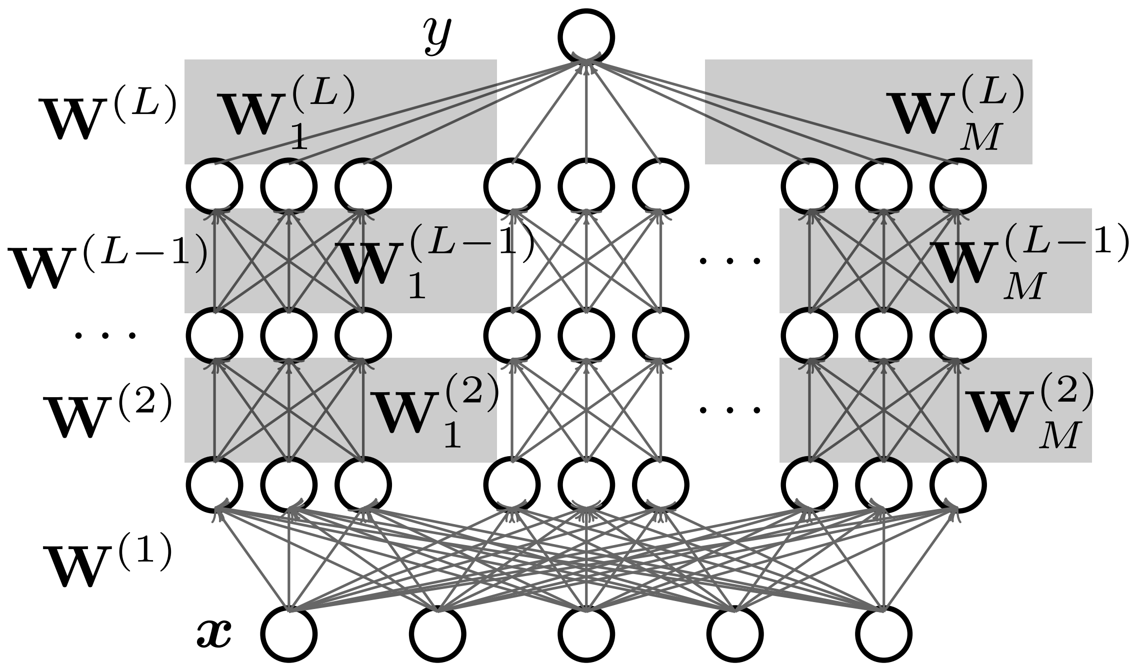
Still slightly unsatisfactory, because…

- Non-typical activation functions / regularization
  - Choice tied to a particular function class
- (Almost) no representation learning
  - Except learning where the knots are
- Not stable when made deeper

Outline

- Motivation
  - Mysteries around deep neural networks
  - Probe it from nonparametric regression angle
- Warm-up
  - 2-layer NN with weight decay vs LAR Splines
- Main results
  - L-layer parallel NN vs Sparse Linear Regression
  - Error bound and discussion
- Proof sketch

L-Layer *Parallel* Neural Networks

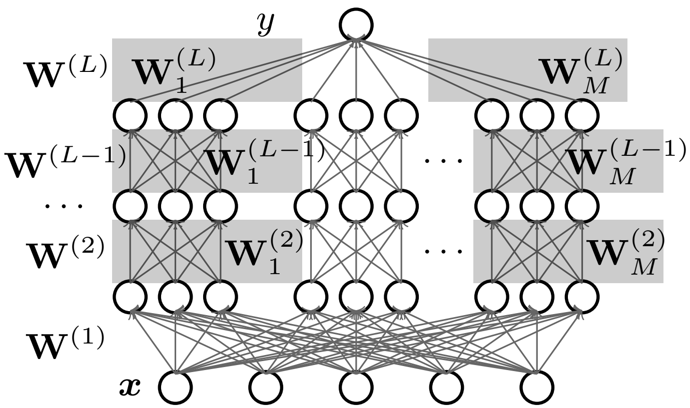


$$\min_{f_j} L(\textstyle\sum_j f_j) + \lambda \sum_{\ell=1}^{L} \sum_{j=1}^{M} \|\mathbf{W}_j^{(\ell)}\|_F^2.$$

(a) Parallel NN with Weight Decay

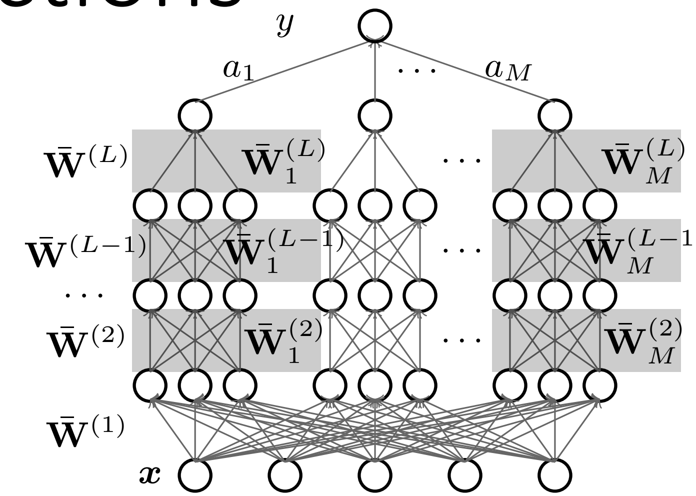
(Ergen&Pilanci, 2021; Haeffele & Vidal, 2017). Also, SqueezeNet, ResNeXT etc.

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



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

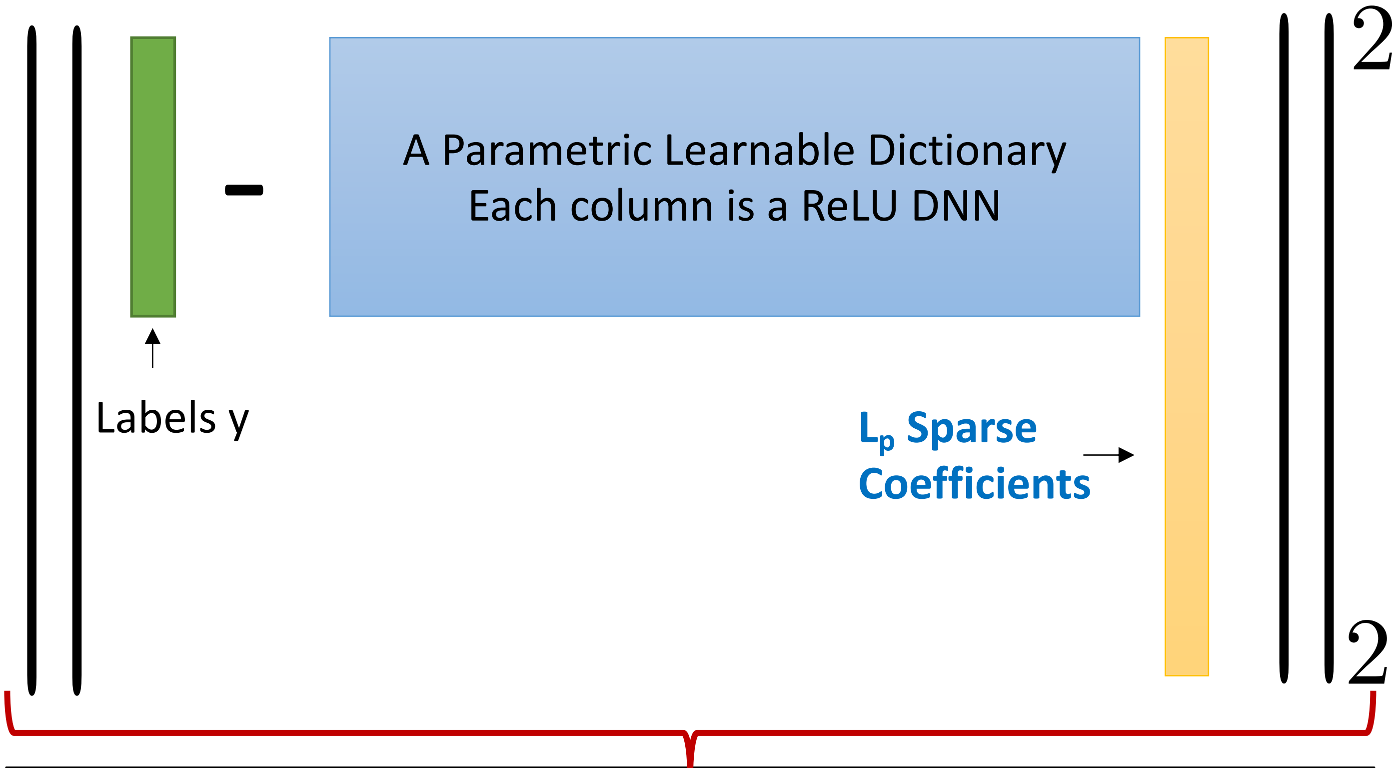
(b) Sparse Regression with Learned Representation

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

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

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

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



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

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

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

Formal setup / notations

- Function classes
  - Bounded Variation class: $BV(m) := \{f : TV(f^{(m)}) < \infty\}.$
  - 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
    - Number of params p >> n in this problem

Comparing to classical nonparametric regression methods

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

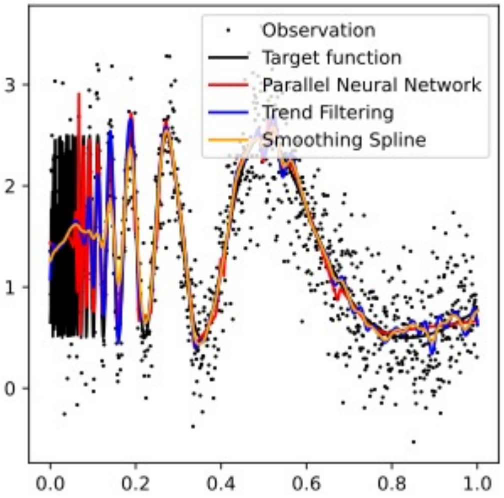
$$[g_1, \ldots, g_M]$$

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

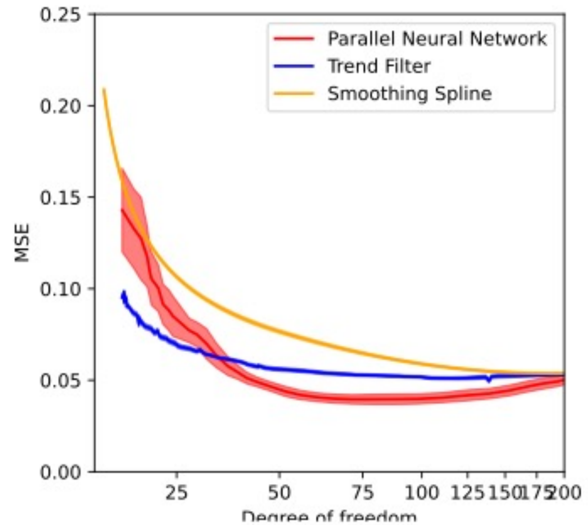
	LAR Splines / Trend filtering	Wavelet smoothing	Parallel DNN
Basis functions	Hard-coded for each order of smoothness	Hard-coded to the chosen wavelets	Parametric and learned from data.
Coefficient vector	L1-sparsity	L1 or L0-sparsity	Lp sparsity (p=2/L)

- 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!

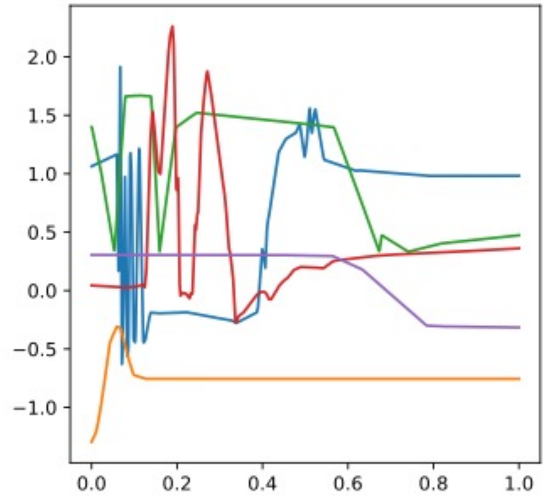
Examples of Functions with Heterogeneous Smoothness



Fitted functions with optimally tuned parameter



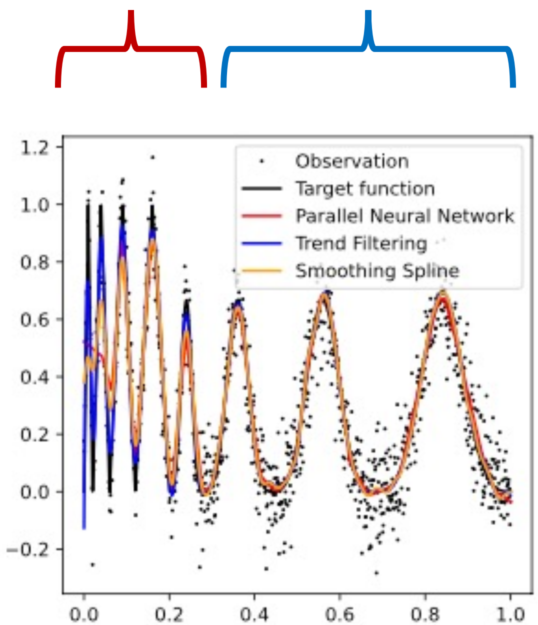
MSE comparison over effective degree of freedom



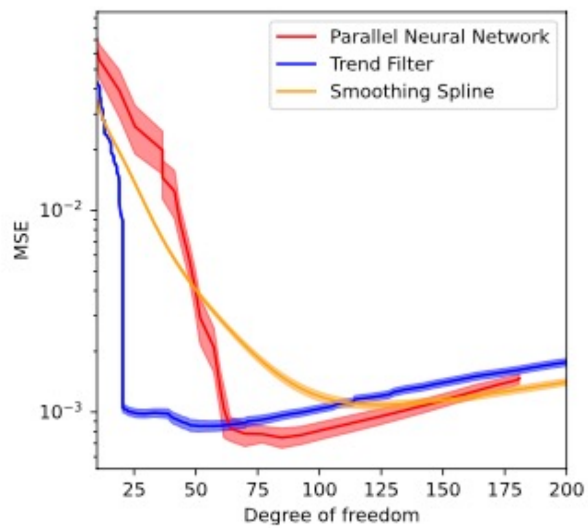
Learned basis functions. Only a handful that are active, i.e. sparsity. Lottery ticket?

Examples of Functions with **even more** Heterogeneous Smoothness

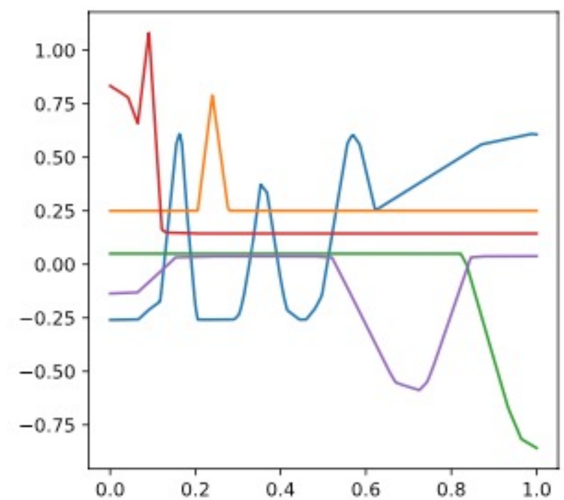
Piecewise Linear Piecewise Cubic



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?

Outline

- Motivation
  - Mysteries around deep neural networks
  - Probe it from nonparametric regression angle
- Warm-up
  - 2-layer NN with weight decay vs LAR Splines
- Main results
  - L-layer parallel NN vs Sparse Linear Regression
  - Error bound and discussion
- Proof sketch

Proof sketch

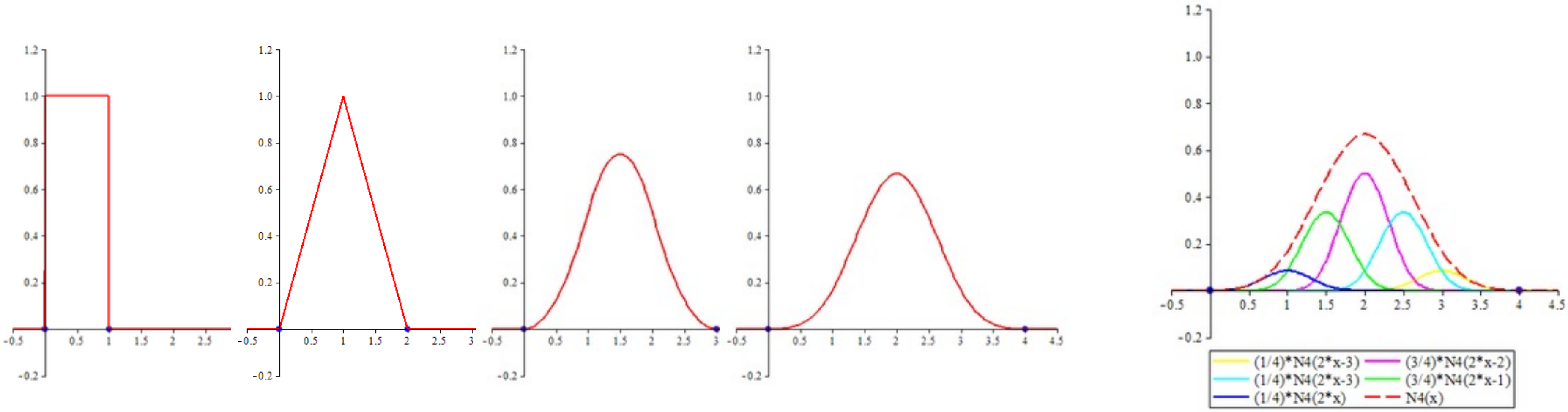
- **Step 1: Proposition 14:** Fast rate in Fixed Design with an unregularized Nullspace

$$\mathrm{MSE}(\hat{f}) = O\Bigg(\underbrace{\inf_{f \in \mathcal{F}} \mathrm{MSE}(f)}_{\text{approximation error}} + \underbrace{\frac{\log \mathcal{N}(\mathcal{F}_{\|}, \delta, \|\cdot\|_\infty) + d(\mathcal{F}_\perp)}{n} + \delta}_{\text{estimation error}} \Bigg)$$

  - Standard self-bounding arguments
  - But need to handle various technical issues

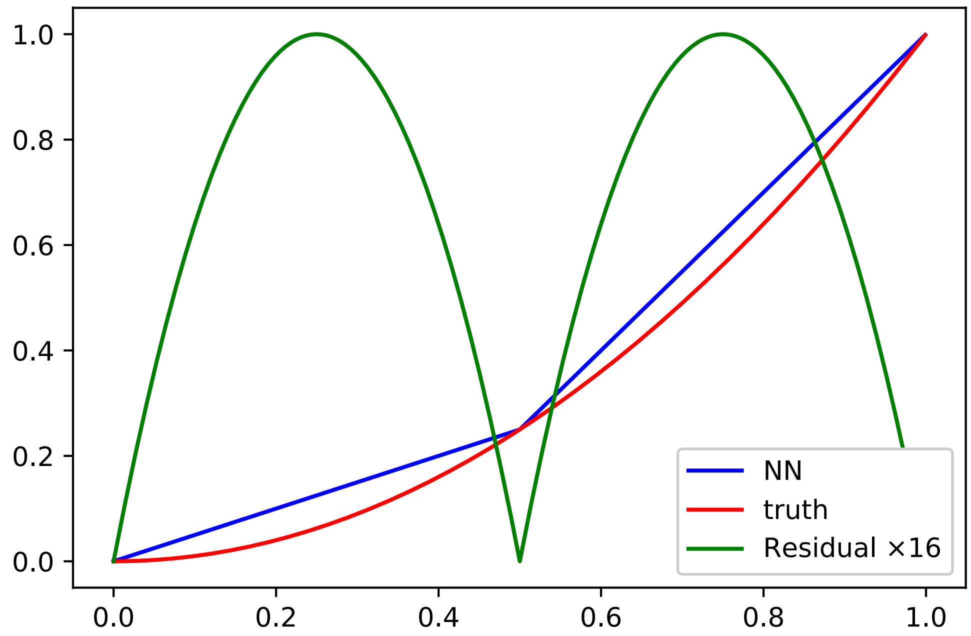
Step 2: Approximation Error Bound

- **Proposition 7:** Each subnetwork can approximate a **cardinal B-spline basis** for all orders, with scaling / shift
  - Techniques of Yarotsky [2017] with some extensions

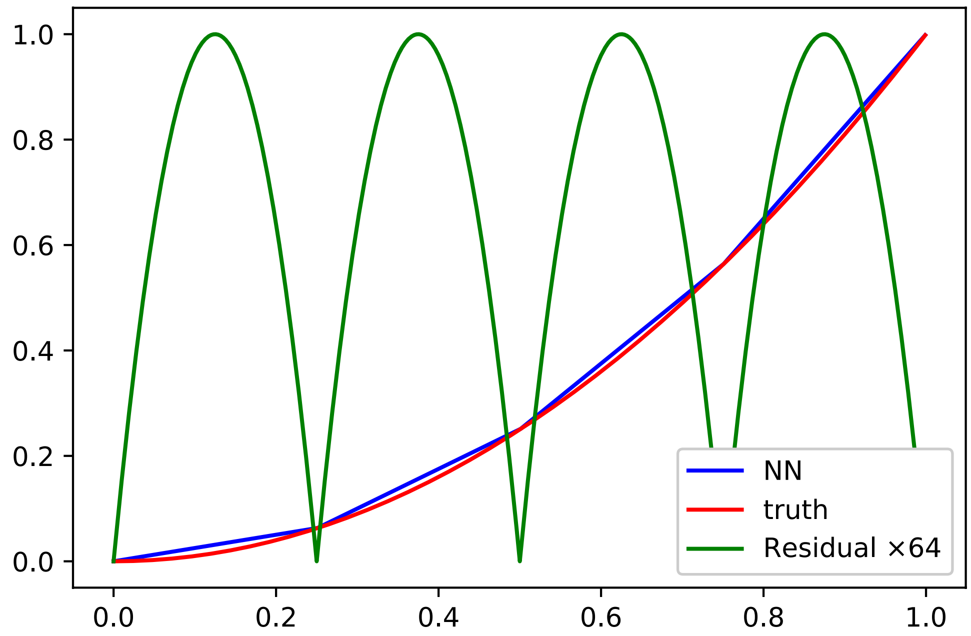


- **Proposition 8:** Sparse combination of cardinal B-spline wavelets approximates all functions in Besov space.
  - Techniques from (Dung, 2011) and (Suzuki, 2019)

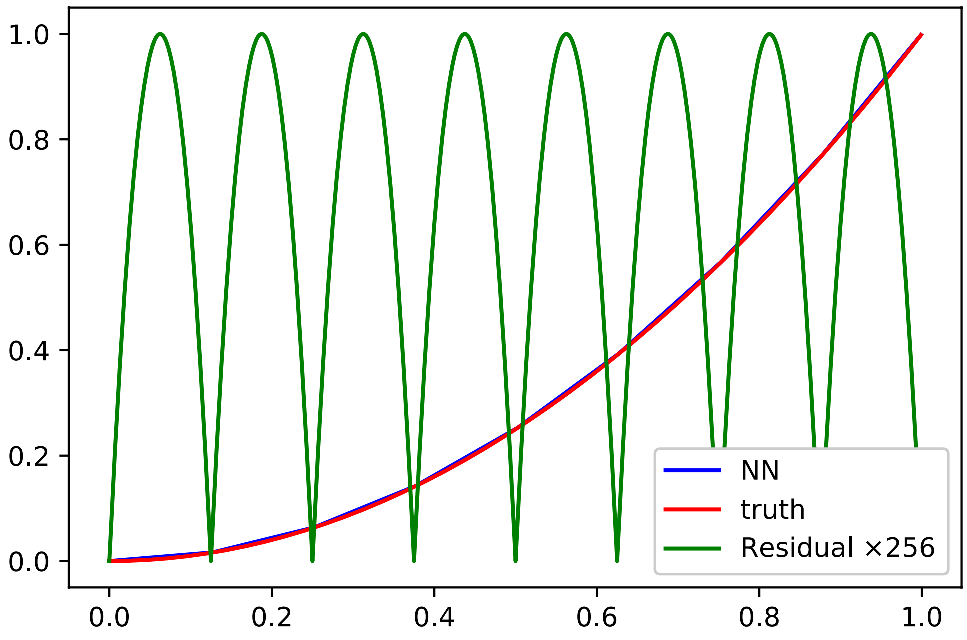
ReLU NN's approximation of x^2 as it gets deeper



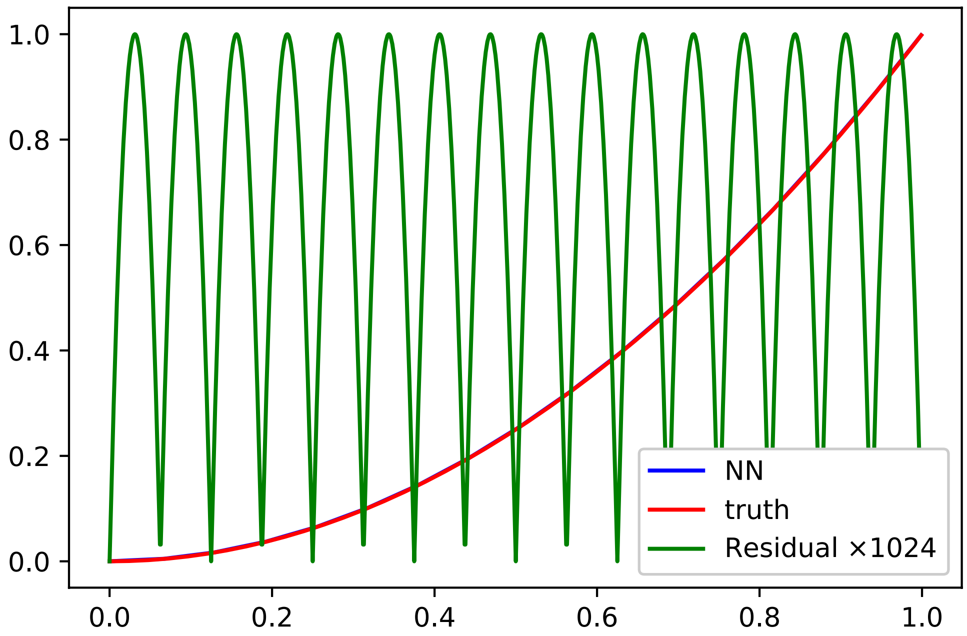
ReLU NN's approximation of x^2 as it gets deeper



ReLU NN's approximation of x^2 as it gets deeper



ReLU NN's approximation of x^2 as it gets deeper



Step 3: Metric Entropy of the Lp norm bounded combinations of ReLU NN

- **Lemma 6.** Bounding covering number of L_p sparse combinations

$$\mathcal{N}(\mathcal{G}, \delta) \lesssim \delta^{-k} \log(1/\delta)$$

$$\mathcal{F} = \left\{ \sum_{i=1}^{M} a_i g_i \middle| g_i \in \mathcal{G}, \|a\|_p^p \leq P, 0 < p < 1 \right\}$$

- Then $\log \mathcal{N}(\mathcal{F}, \epsilon) \lesssim kP^{\frac{1}{1-p}} (\delta/c_3)^{-\frac{p}{1-p}} \log(c_3 P/\delta)$

- Note the independence to the number of subnetworks.
It can be **arbitrarily overparameterized**!
- But our bound requires only M to be mildly over-parameterized.

Summary of take-home messages

- Separation from kernel methods
- Depth advantage
- Adaptivity advantage
  - Tuning weight decay is all that is needed
- Implicit sparsity in a learned dictionary space
  - Computational benefits in deployment time?

Future work

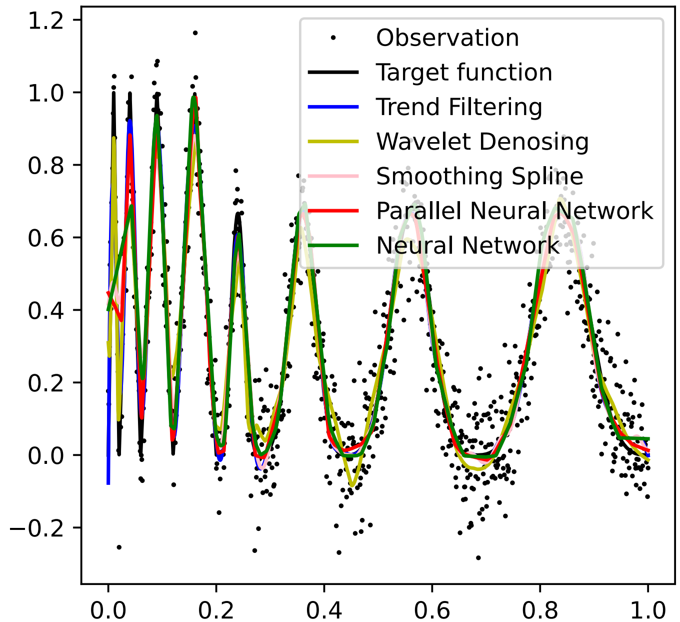
- Formalizing the sub-region local adaptivity
- Non-parallel NNs with weight decay
- Locally adaptivity in transformed space, e.g., Fourier domain (CNNs?)
- Multi-task setting ⇔ Dictionary learning?
- Biological neural science interpretation (Michael Beyeler has some thoughts)

Thank you for your attention!

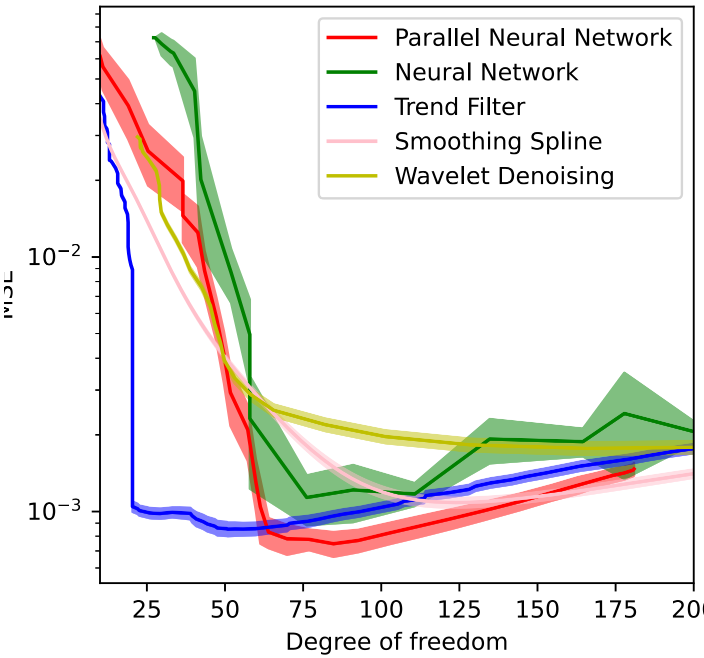
- References:
  - Zhang and W. (2022) "Deep Learning Meets Nonparametric Regression" https://arxiv.org/abs/2204.09664
  - Suzuki (ICLR'2019) https://arxiv.org/abs/1810.08033
  - Parhi and Nowak (JMLR'21) https://jmlr.org/papers/v22/20-583.html

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

Supplementary slide: Comparing to Wavelets



(d) “Vary”, DoF=50.



(e) MSE versus DoF