Supervised Learning In Banach Space

Approximation Theory For Operators

Andrew Stuart

Computing and Mathematical Sciences
California Institute of Technology

AFOSR, ARL, NSF, ONR

$\star\star\star$

IPAM Closing Workshop

June 8^{th} 2020

Collaborators

- K Bhattacharya, B Hosseini, NB Kovachki and AMS
  Model Reduction And Neural Networks For Parametric PDEs
  arXiv:2005.03180
- NH Nelsen and AMS
  The Random Feature Model For Input-Output Maps Between Banach Spaces
  arXiv:2005.10224
- Z Li, NB Kovachki, K Azizzadenesheli,
  B Liu, K Bhattacharya, AMS and A Anandkumar
  Neural Operator: Graph Kernel Network for Partial Differential Equations
  arXiv:2003.03485

Overview

Supervised Learning

The Talk In A Couple Of Slides

Model Reduction Approach

Conclusions and References

Random Features Approach

Graph Kernel Approach

The Setting

The Problem

Setting

$$\begin{aligned} \text{Input-Output Map:} \quad & \Psi^\dagger : \mathcal{X} \mapsto \mathcal{Y}. \\ \text{Regression:} \quad & \mathcal{X} = \mathbb{R}^q, \mathcal{Y} = \mathbb{R}^r. \\ \text{Classification:} \quad & \mathcal{X} = \mathbb{R}^q, \mathcal{Y} = \{1, \ldots, K\}. \end{aligned}$$

Parametric Approximation Of $\Psi^\dagger$

$$\begin{aligned} \text{Data:} \quad & \{x_n, y_n\}_{n=1}^N, \quad y_n = \Psi^\dagger(x_n). \\ \text{Parameter Space:} \quad & \Theta \subseteq \mathbb{R}^p. \\ \text{Approximation Class:} \quad & \Psi : \mathcal{X} \times \Theta \mapsto \mathcal{Y}. \\ \text{Goal:} \quad & \theta^* \in \Theta : \Psi(\cdot\,; \theta^*) \approx \Psi^\dagger. \end{aligned}$$

[1] Y LeCun, Y Bengio, G Hinton. (deep learning and applications)

Training

μ a probability measure on $\mathcal{X}$; $\quad \ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ a distance.

Perfect Setting

$$\text{Objective Function:} \quad \mathsf{J}(\theta) = \mathbb{E}^{x \sim \mu} \ell\Big(\Psi(x;\theta), \Psi^\dagger(x)\Big).$$

$$\text{Minimizer:} \quad \theta^\star = \operatorname{argmin}_{\theta \in \Theta} \mathsf{J}(\theta).$$

Let $x_n \overset{i.i.d.}{\sim} \mu$. and $y_n = \Psi^\dagger(x_n)$.

Data-Driven Setting

$$\text{Objective Function:} \quad \mathsf{J}^N(\theta) = \frac{1}{N} \sum_{n=1}^{N} \ell\Big(\Psi(x_n;\theta), y_n\Big),$$

$$\text{Minimizer:} \quad \theta^{\star,N} = \operatorname{argmin}_{\theta \in \Theta} \mathsf{J}^N(\theta).$$

Testing

Perfect Setting

$$\text{Error Measure:} \quad e_N = \mathbb{E}^{x\sim\mu}\ell\Big(\Psi(x;\theta^{\star,N}), \Psi^\dagger(x)\Big).$$

Let $x'_m \overset{i.i.d.}{\sim} \mu'$. and $y'_m = \Psi^\dagger(x'_m)$. $\{x'_m\}_{m=1}^M \perp\!\!\!\perp \{x_n\}_{n=1}^N$.

Data-Driven Setting

$$\text{Error Measure:} \quad e_{N,M} = \frac{1}{M}\sum_{m=1}^{M}\ell\Big(\Psi(x'_m;\theta^{\star,N}), y'_m\Big).$$

This talk: $\mu' = \mu$; but $\mu \neq \mu'$ also relevant.

Neural Networks

Deep And/Or Wide

$$
\begin{aligned}
v_0 &= Ax + a, \\
v_{k+1} &= \sigma(W_k v_k + \omega_k), \quad k = 0, 1, \cdots, K-1, \\
\Psi(x;\theta) &= Bv_K + b.
\end{aligned}
$$

Notation

$$\sigma(z) = \max(0, z) \qquad \text{(Nemytskii)},$$

$$\theta = (A, a, B, b) \bigcup_{k=1}^{K} \{W_k, \omega_k\},$$

[2] I Goodfellow, Y Bengio, A Courville. (deep learning text book)

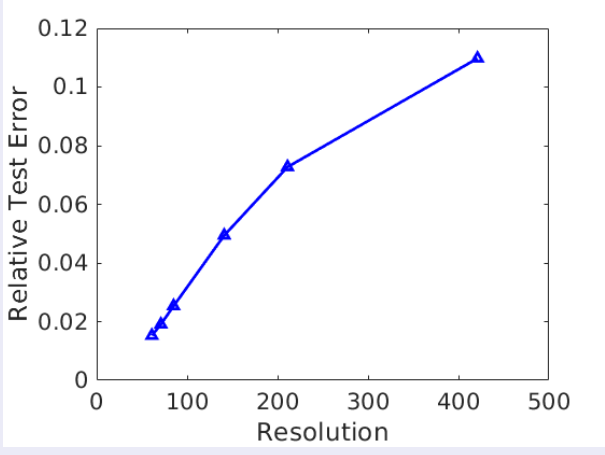
The Talk In A Couple Of Slides

Previous work at PDE/neural networks interface:

[3] M Raissi, P Perdikaris, GE Karniadakis;

[4] Weinan E and B Yu.

Example What Goes Wrong If You Discretize And Then Apply Standard Neural Network Algorithms



[5a] Y Zhu and N Zabaras; see also [5b] Y Khoo, J Lu and L Ying

The Problem (Revisited)

Key Idea

Design Architecture On Banach Space Then Discretize

Setting

Input-Output Map: $\Psi^\dagger : \mathcal{X} \mapsto \mathcal{Y}$, Separable Banach Spaces

Data: $\{x_n, y_n\}_{n=1}^N$, $y_n = \Psi^\dagger(x_n)$, $x_n \overset{i.i.d.}{\sim} \mu$.

Goal

Parameter Space $\Theta \subseteq \mathbb{R}^p$

Operator Class: $\Psi : \mathcal{X} \times \Theta \mapsto \mathcal{Y}$

Operator Approximation: $\Psi(\cdot; \theta^\star) \approx \Psi^\dagger$

Model Reduction Approach

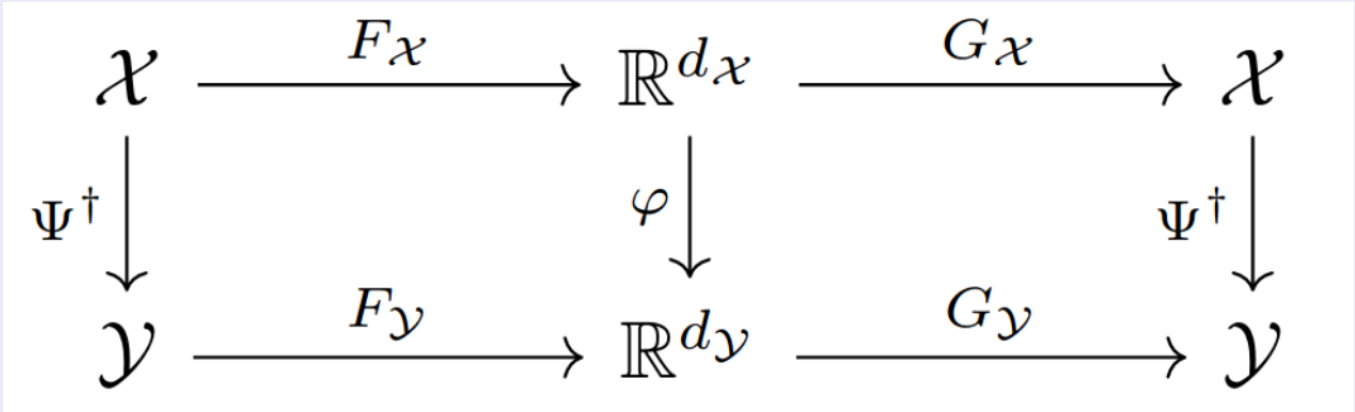
K Bhattacharya, B Hosseini, NB Kovachki and AMS

Model Reduction And Neural Networks For Parametric PDEs

arXiv:2005.03180

The Idea

In A Picture



In Equations

$$G_{\mathcal{X}} \circ F_{\mathcal{X}} \approx I_{\mathcal{X}},$$
$$G_{\mathcal{Y}} \circ F_{\mathcal{Y}} \approx I_{\mathcal{Y}},$$
$$G_{\mathcal{Y}} \circ \varphi \circ F_{\mathcal{X}} \approx \Psi^\dagger.$$

Principal Components Analysis

Required Information

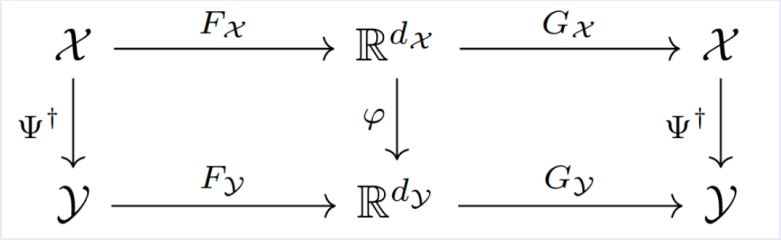
$$\begin{aligned} \text{Hilbert Space} \quad & \mathcal{H} \\ \text{Probility Measure:} \quad & \pi \\ \text{Mean:} \quad & m = \mathbb{E}^{h\sim\pi} h \\ \text{Covariance:} \quad & C = \mathbb{E}^{h\sim\pi}(h-m)\otimes(h-m) \end{aligned}$$

Projection

$$\begin{aligned} \text{EVP:} \quad & C\psi_j = \lambda_j \psi_j \\ \text{F}: \mathcal{H} \mapsto \mathbb{R}^d \quad & Fh = \{v_j := \langle h, \psi_j \rangle\}_{j=1}^d \\ \text{G}: \mathbb{R}^d \mapsto \mathcal{H} \quad & Gv = \sum_{j=1}^{d} v_j \psi_j. \end{aligned}$$

The Approximations

PCA ($\mathcal{X}, \mathcal{Y}$ Hilbert spaces)



$$\varphi := F_{\mathcal{Y}} \circ \Psi^\dagger \circ G_{\mathcal{X}},$$
$$\Psi_{PCA} := G_{\mathcal{Y}} \circ \varphi \circ F_{\mathcal{X}}.$$

Neural Networks

$$\psi \approx \varphi,$$
$$\Psi_{NN} := G_{\mathcal{Y}} \circ \psi \circ F_{\mathcal{X}}.$$

Main Theorem

Theorem

For any $\epsilon > 0$, there are dimensions $(d_{\mathcal{X}}, d_{\mathcal{Y}})(\epsilon)$, a requisite amount of data $N = N(d_{\mathcal{X}}, d_{\mathcal{Y}})$, and a zero-extended neural network ψ depending on $\epsilon, d_{\mathcal{X}}, d_{\mathcal{Y}}$ such that

$$\mathbb{E}_{\{x_j\}\sim\mu}\mathbb{E}_{x\sim\mu}\|\Psi_{NN}(x) - \Psi^{\dagger}(x)\|_{\mathcal{Y}}^2 < \epsilon.$$

[6] L Zwald, O Bousquet, G Blanchard (used in proof)

[7] D Yarotsky (used in proof)

Related work:

[8] G Kutyniok, P Petersen M Raslan and R Schneider ; (finite dimensional nn approximation)

[9] C Schwab, J Zech; (finite dimensional nn approximation)

[10] A Chkifa, A Cohen, R DeVore and C Schwab; (parametric pde approximation)

[11] RA DeVore. (parametric pde approximation)

Example

Elliptic PDE

$$-\nabla \cdot (a \nabla u) = f, \quad z \in D$$
$$u = g, \quad z \in \partial D.$$

Operators Of Interest

Linear (Boundary) $\Psi^\dagger : g \in H^{\frac{1}{2}}(\partial D) \mapsto u \in H^1(D)$;

Linear (Forcing) $\Psi^\dagger : f \in L^2(D) \mapsto u \in H^1(D)$;

Nonlinear $\Psi^\dagger : a \in L^\infty(D) \mapsto u \in H^1(D)$.

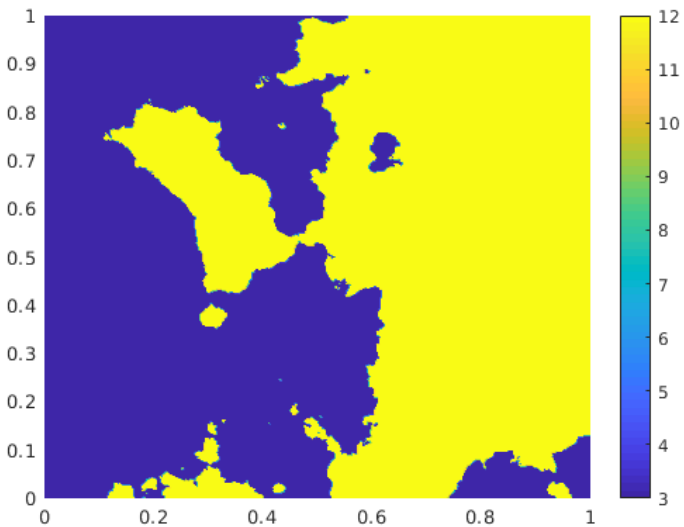
Distance

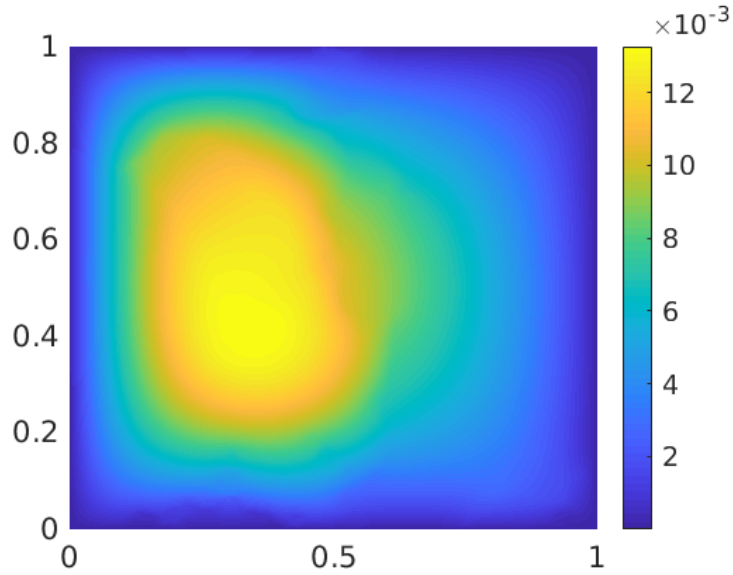
$$\ell(v, u) = \|v - u\|_{L^2(D)} / \|u\|_{L^2(D)}.$$

Example

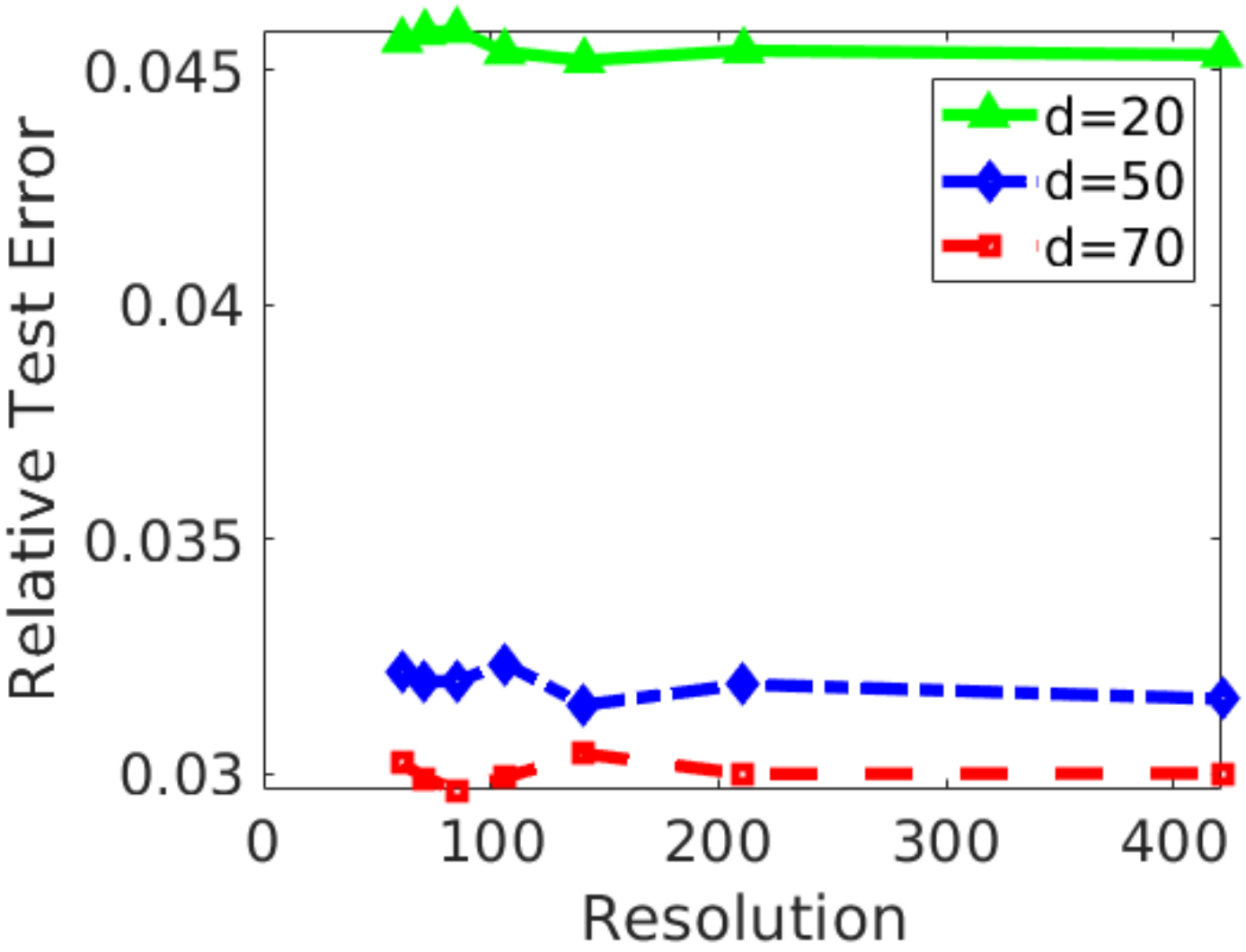
Input-Output

$$\text{Input:} \quad a \in L^{\infty}(D) \quad (\textit{Left}),$$
$$\text{Output:} \quad u \in H^{1}(D). \quad (\textit{Right}),$$





Example


0.045
0.04
0.035
0.03
Relative Test Error
0
100
200
300
400
Resolution
d=20
d=50
d=70

Conclusions and References

Conclusions

1. Neural networks: empirical success in function approximation.
2. Typically:
   - regression:$\mathbb{R}^m \mapsto \mathbb{R}^n$;
   - classification:$\mathbb{R}^m \mapsto \{1, \ldots, K\}$.
3. We consider: $\mathcal{X} \mapsto \mathcal{Y}$, $\mathcal{X}, \mathcal{Y}$ function spaces.
4. Bad: Discretize then apply 2.
5. Good: Conceive of architecture then discretize.
6. Purely data-driven ("equation-free").
7. Applications: PDEs
8. Applications: Other?

References

[1] Y LeCun, Y Bengio, G Hinton .
Deep Learning
Nature 2015.

[2] I Goodfellow, Y Bengio, A Courville
Deep Learning
MIT Press 2016

[3] M Raissi, P Perdikaris, GE Karniadakis
Physics-informed neural networks...
J. Comp. Phys. 2019.

[4] Weinan E and B Yu
The Deep Ritz Method...
Communications in Mathematics and Statistics 2018.

[5a] Y Zhu and N Zabaras
Bayesian Deep Convolutional Encoder-Decoder Networks...
J. Comp. Phys. 2018.

[5b] Y Khoo, J Lu, L Ying
Solving parametric PDE problems with artificial neural networks
arXiv:1707.03351,

[6] L Zwald, O Bousquet, G Blanchard
Statistical properties of kernel principal component analysis...
International Conference on Computational Learning Theory, Springer, 2004.

References

[7] D Yarotsky

Error bounds for approximations with deep ReLU networks

Neural Networks 2017.

[8] G Kutyniok, P Petersen M Raslan and R Schneider

A theoretical analysis of deep neural networks and parametric PDEs

arXiv:1904.00377

[9] C Schwab, J Zech

Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ

Analysis and Applications 2019

[10] A Chkifa, A Cohen, R DeVore and C Schwab

Sparse adaptive Taylor approximation algorithms for parametric and stochastic elliptic PDEs

ESAIM: Mathematical Modelling and Numerical Analysis 2013

[11] RA DeVore

The Theoretical Foundation of Reduced Basis Methods

Model Reduction and Approximation, SIAM, 2014

[12] A Rahimi, B Recht

Uniform approximation of functions with random bases

46th Annual Allerton Conference on Communication, Control, and Computing, 2008

Random Features Approach

NH Nelsen and AMS

The Random Feature Model For Input-Output Maps Between Banach Spaces

arXiv:2005.10224

The Architecture

Random Feature Mapping

Probability Space: $(\Gamma, \mathcal{F}, \nu)$

Random Feature: $\psi(\cdot\,;\gamma) : \mathcal{X} \to \mathcal{Y}, \quad \gamma \sim \nu$

Random Feature Model

Parameter Space : $\Theta \subseteq \mathbb{R}^m$

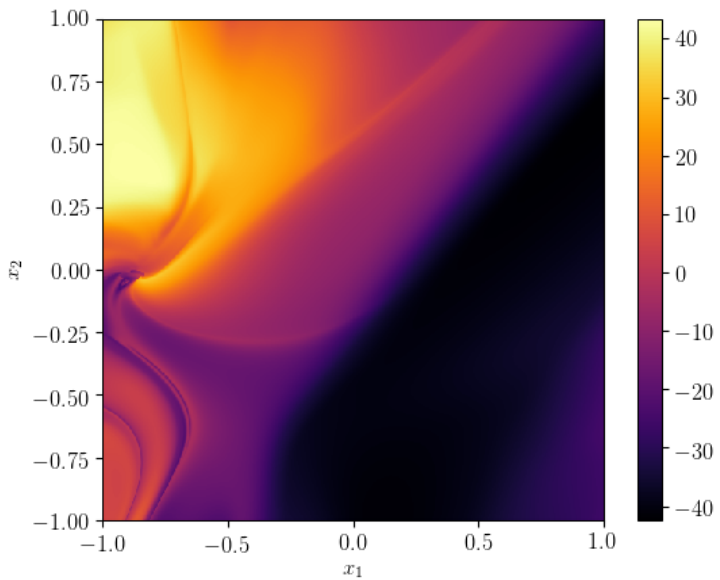
Architecture : $\Psi_m(x;\theta) := \dfrac{1}{m}\displaystyle\sum_{j=1}^{m} \theta_j \psi(x;\gamma_j)\,, \quad \gamma_j \sim \nu$ i. i. d.

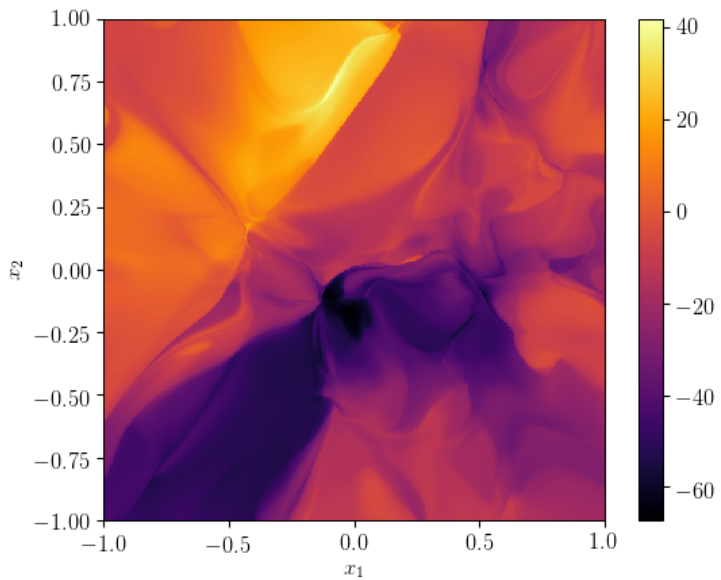
[12] A Rahimi and B Recht. (random features and approximation theory)

Example (Finite Dimensions For Initial Illustration)

Using ResNet As Random Features

$$\begin{aligned}
\text{Input Space:} \quad & \mathcal{X} = \mathbb{R}^2, A \in \mathbb{R}^{1\times 2} \\
\text{Output Space:} \quad & \mathcal{Y} = \mathbb{R}, a \in \mathbb{R} \\
\text{Weights:} \quad & W \in C([0,1];\mathbb{R}^{2\times 2}),\ \omega \in C([0,1];\mathbb{R}^2), \\
\text{Random Evolution:} \quad & \dot{v}(t) = \sigma\big(W(t)v(t) + \omega(t)\big), v(0) = x, \\
\text{Random Features:} \quad & \psi(x;\gamma) = Av(1) + a, \quad \gamma = (W, \omega, A, a).
\end{aligned}$$





Reproducing Kernel Hilbert Space (RKHS)

Kernel From Features

$$\begin{aligned}
\text{Kernel:} \quad & k : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y}, \mathcal{Y}), \\
\text{Features:} \quad & k(x, z) := \mathbb{E}^{\gamma \sim \nu} \Big(\psi(x; \gamma) \otimes \psi(z; \gamma) \Big), \\
\text{RKHS:} \quad & \mathcal{H}_k \subset L^2_\mu(\mathcal{X}; \mathcal{Y}),
\end{aligned}$$

Integral Representation

$$\begin{aligned}
f(z) &= \int \theta(\gamma) \psi(z; \gamma) \nu(d\gamma), \\
\theta(\gamma) &= \langle f, \psi(\cdot; \gamma) \rangle_{\mathcal{H}_k}
\end{aligned}$$

From Nonparametric to Parametric Regression

Gaussian Process Regression

$$\text{Penalized LSQ:} \quad \mathsf{I}(F) := \sum_{n=1}^{N} \frac{1}{2} \|y_n - F(x_n)\|_{\mathcal{Y}}^2 + \frac{\lambda}{2} \|F\|_{\mathcal{H}_k}^2,$$

$$\text{Regression:} \quad F^* = \operatorname{argmin}_{F \in \mathcal{H}_k} \mathsf{I}(F), \qquad F^*(x) := \sum_{\ell=1}^{N} k(x, x_\ell) \beta_\ell.$$

Link To Random Features Model

$$\text{Parametric:} \quad k(x, z) \approx k^{(m)}(x, z) = \frac{1}{m} \sum_{j=1}^{m} \psi(x; \gamma_j) \otimes \psi(z; \gamma_j),$$

$$\text{Objective:} \quad \mathsf{J}^\lambda(\theta) := \sum_{n=1}^{N} \frac{1}{2} \|y_n - \frac{1}{m} \sum_{\ell=1}^{m} \theta_\ell \psi(x_n; \gamma_\ell)\|_{\mathcal{Y}}^2 + \frac{\lambda}{2m} \|\theta\|_2^2,$$

$$\text{Regularized RFM:} \quad \theta^* := \operatorname{argmin}_{\theta \in \mathbb{R}^m} \mathsf{J}^\lambda(\theta).$$

Example

Burgers Equation

$$\partial_t u + \partial_s(u^2/2) = \delta \partial_{ss}^2 u, \quad (s,t) \in \mathbb{T}^1 \times (0,1],$$
$$u|_{t=0} = u_0, \quad s \in \mathbb{T}^1.$$

Operator Of Interest

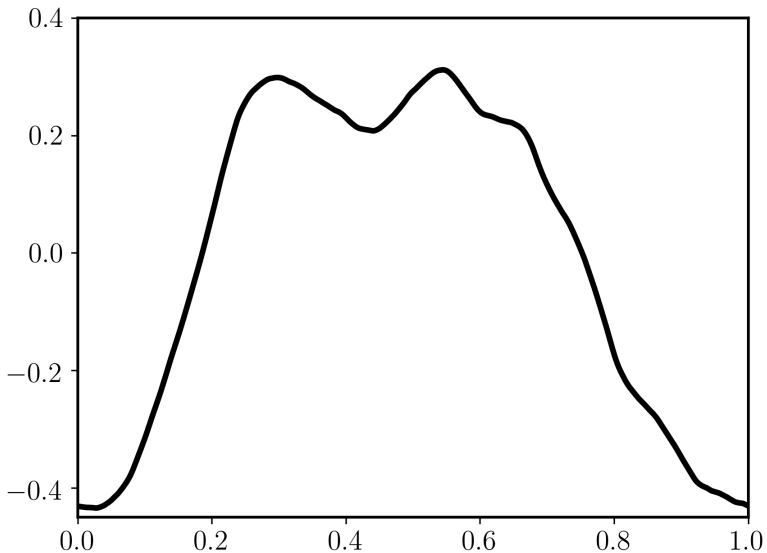
Nonlinear $\quad \Psi^\dagger : u_0 \in L^2(\mathbb{T}^1) \mapsto u|_{t=1} \in H^r(\mathbb{T}^1).$

Example

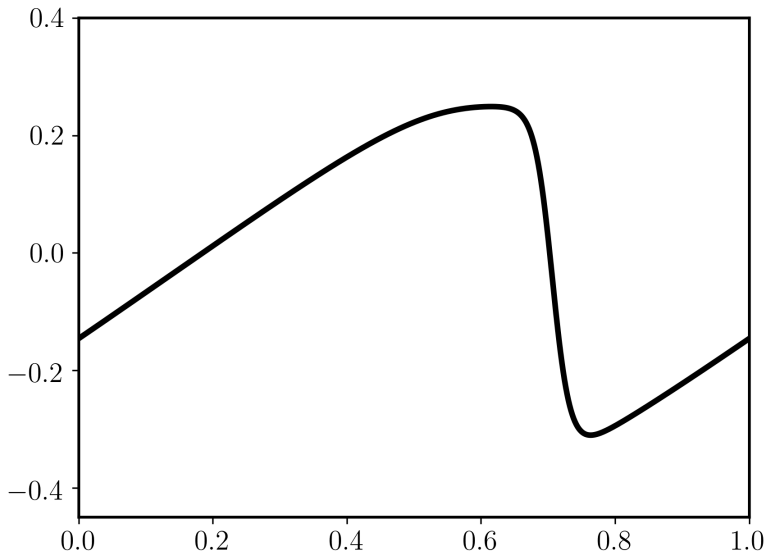
Input-Output

$$\text{Input:} \quad u_0 \in L^2(\mathbb{T}^1), \quad (\textit{Left})$$

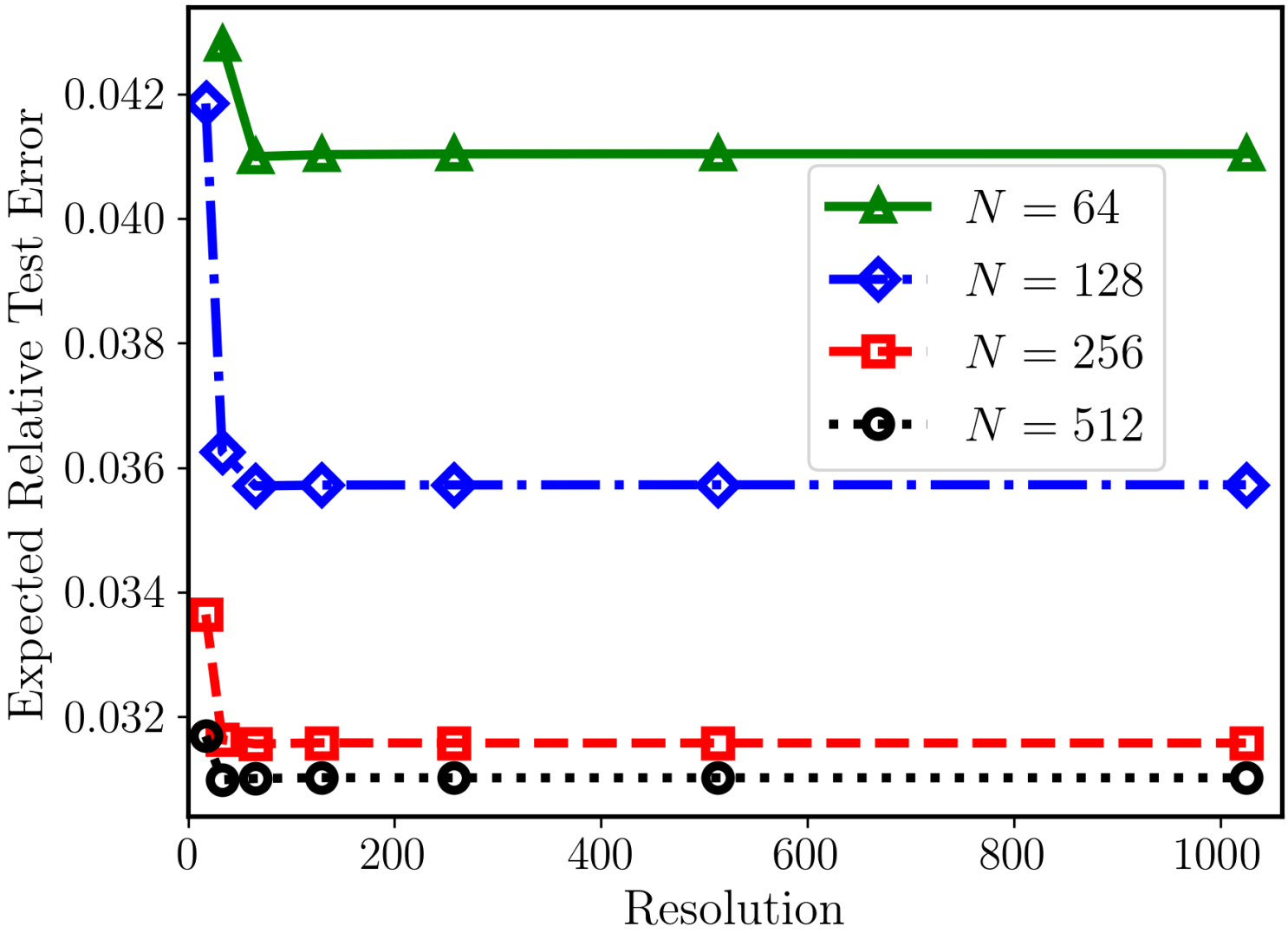
$$\text{Output:} \quad u|_{t=1} \in H^r(\mathbb{T}^1). \quad (\textit{Right})$$



say



Example


Expected Relative Test Error
0.042
0.040
0.038
0.036
0.034
0.032
0
200
400
600
800
1000
Resolution
$N = 64$
$N = 128$
$N = 256$
$N = 512$

Graph Kernel Approach

Z Li, NB Kovachki, K Azizzadenesheli,
B Liu, K Bhattacharya, AMS and A Anandkumar

Neural Operator: Graph Kernel Network for Partial Differential Equations

arXiv:2003.03485

Integral Kernel Neural Networks

Deep And/Or Wide

$$
\begin{aligned}
f(s) &= (s, x(s), x_\epsilon(s), \nabla_s x_\epsilon(s))^T, \\
v_0(s) &= Af(s) + a, \\
v_{k+1}(s) &= \sigma\Big(Wv_k(s) + \int k(s, r, x(s), x(r); \vartheta) v_k(r) \lambda(dr)\Big), \\
\Psi(x; \theta) &= Bv_K(s) + b.
\end{aligned}
$$

In Abstract Notation

$$
\begin{aligned}
k(\cdot, \vartheta) &\in \{\varphi : \mathbb{R}^{2d+2} \mapsto \mathbb{R}^{p \times p}\}, \\
\theta &= (A, a, B, b, W, \vartheta).
\end{aligned}
$$

Example

Elliptic PDE

$$-\nabla \cdot (a \nabla u) = f, \quad z \in D$$
$$u = g, \quad z \in \partial D.$$

Operators Of Interest

Linear (Boundary) $\Psi^\dagger : g \in H^{\frac{1}{2}}(\partial D) \mapsto u \in H^1(D);$

Linear (Forcing) $\Psi^\dagger : f \in L^2(D) \mapsto u \in H^1(D);$

Nonlinear $\Psi^\dagger : a \in L^\infty(D) \mapsto u \in H^1(D).$

Distance

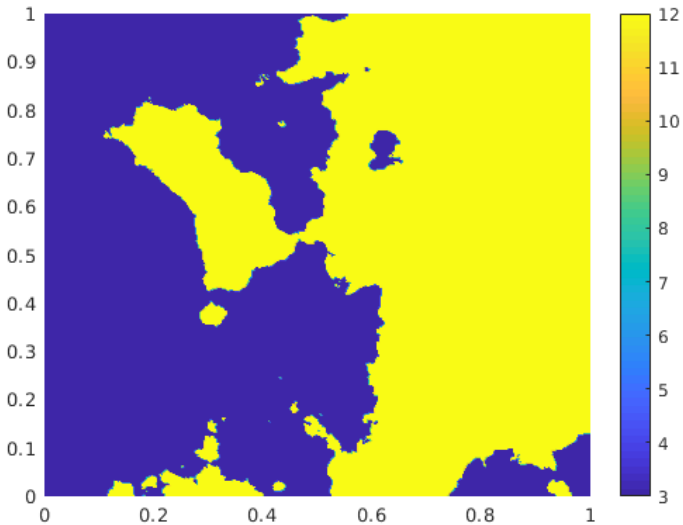
$$\ell(v, u) = \|v - u\|_{L^2(D)} / \|u\|_{L^2(D)}.$$

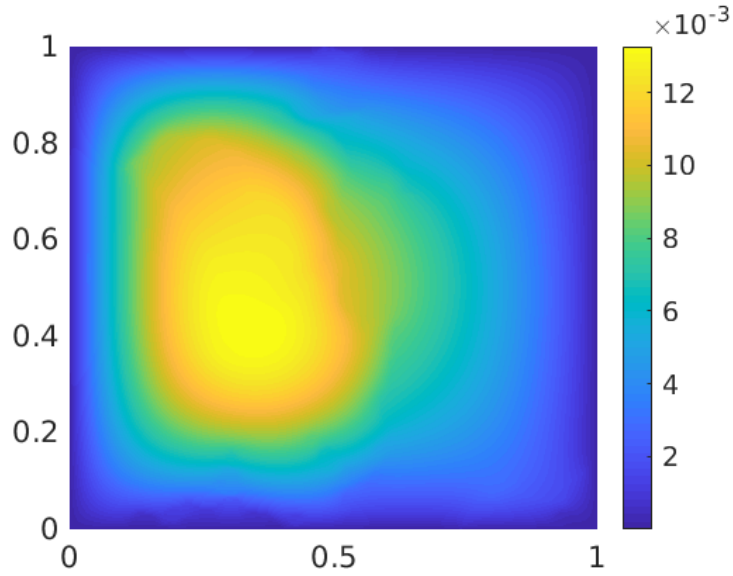
Example

Input-Output Nonlinear

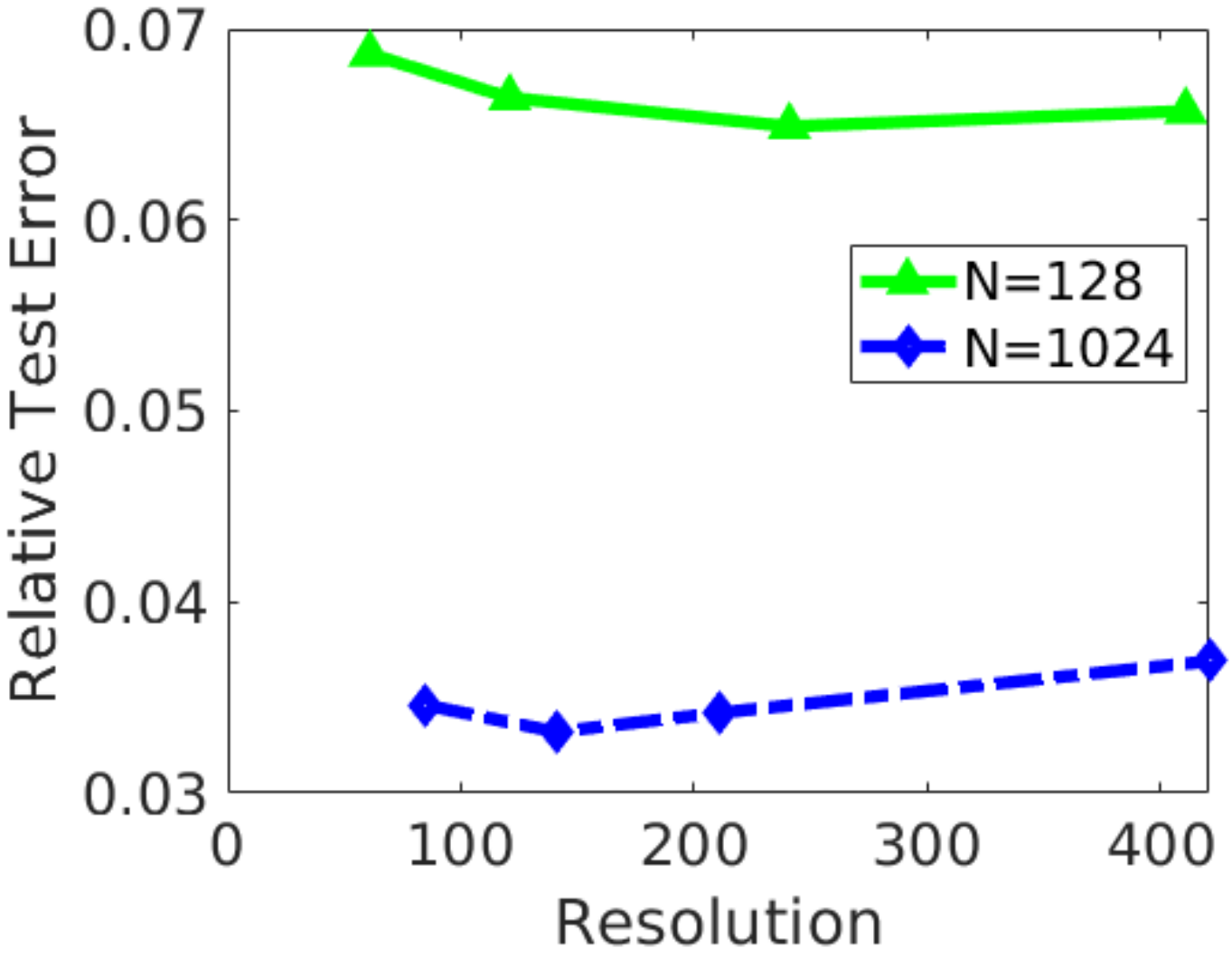
Input: $a \in L^{\infty}(D)$, *(Left)*

Output: $u \in H^{1}(D)$. *(Right)*





Example


0.07
0.06
0.05
0.04
0.03
Relative Test Error
0
100
200
300
400
Resolution
N=128
N=1024

Example

Input-Output Linear (Green's Function $D = [0, 1]$)

$$\text{Input:} \quad f \in L^2(D),$$
$$\text{Output:} \quad u \in H^1(D).$$