Benchmarking Operator Learning with Simple and Interpretable Kernel Methods

Pau Batlle[1], Matthieu Darcy[1], Bamdad Hosseini[2], Houman Owhadi[1]

[1]California Institute of Technology [2]University of Washington

Table of Contents

The operator learning problem

Let $\{u_i, v_i\}_{i=1}^N$ be N elements of $\mathcal{U} \times \mathcal{V}$ such that

$$\mathcal{G}^\dagger(u_i) = v_i, \quad \text{for } i = 1, \dots, N\,.$$

Operator learning problem: general version

Given the data $\{u_i, v_i\}_{i=1}^N$ approximate $\mathcal{G}^\dagger$.

The operator learning problem: finite dimensional version

In practice, we do not have access to u_i, v_i but to pointwise values:

$$\phi : u \mapsto \left(u(x_1), u(x_2), \dots, u(x_m)\right)^T \quad \text{and} \quad \varphi : v \mapsto \left(v(y_1), v(y_2), \dots, v(y_n)\right)^T .$$

Operator learning problem II

Given the data $\{\phi(u_i), \varphi(v_i)\}_{i=1}^N$ approximate $\mathcal{G}^\dagger$.

More generally, ϕ and φ can be bounded linear operators.

In this talk

Past work has focused on Operator Neural Networks[1][2][3] that generalize Neural Networks to functional inputs and outputs. However they have not been benchmarked against simpler methods.

Our contribution

We propose a family of kernel based-methods that are **simple, fast** and **competitive in accuracy.** The methods are natural benchmarks for more complex methods.

---

[1] Zongyi Li et al. *Fourier Neural Operator for Parametric Partial Differential Equations*. 2020.

[2] Kaushik Bhattacharya et al. *Model Reduction and Neural Networks for Parametric PDEs*. 2021.

[3] Lu Lu et al. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229.

Table of Contents

Diagram summary

Summary of our method

Given the data $\{\phi(u_i), \varphi(v_i)\}_{i=1}^N$ our method to approximate $\mathcal{G}^\dagger$:

$$\mathcal{G}^\dagger(u_i) = v_i, \quad \text{for } i = 1, \ldots, N\,.$$

can be summarized in two steps:

1. Define the reconstructions ψ and χ as the optimal recovery map.
2. Approximate the function $f^\dagger$ using a kernel method.

$$\begin{tikzcd}
\mathcal{U} \arrow[r, "\mathcal{G}^\dagger"] \arrow[d, shift left, "\phi\ \text{Measurement}"] & \mathcal{V} \arrow[d, shift left, "\text{Measurement}\ \varphi"] \\
\mathbb{R}^m \arrow[r, "f^\dagger"] \arrow[u, shift left, "\text{Reconstruction}\ \psi"] & \mathbb{R}^n \arrow[u, shift left, "\text{Reconstruction}\ \chi"]
\end{tikzcd}$$

Optimal recovery

We will assume that $\mathcal{U}$ and $\mathcal{V}$ are RKHSs arising from kernels Q and K respectively. The reconstruction operators are defined as optimal recovery maps

$$\psi(\phi(u)) := \arg\min_{w\in\mathcal{U}} \|w\|_Q \quad \text{s.t.} \quad \phi(w) = \phi(u),$$
$$\chi(\varphi(v)) := \arg\min_{w\in\mathcal{V}} \|w\|_K \quad \text{s.t.} \quad \varphi(w) = \varphi(v),$$

The maps are the minmax optimal recovery of u and v respectively[4]. Optimal recovery maps can be expressed in closed form using standard representer theorems for kernel interpolation:

$$\psi(\phi(u))(x) = Q(x, X)Q(X, X)^{-1}\phi(u) \quad \text{and} \quad \chi(\varphi(v))(y) = K(y, Y)K(Y, Y)^{-1}\varphi(v).$$

[4] Houman Owhadi and Clint Scovel. *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2019.

Recovery of $f^\dagger$

Once the reconstruction operators ψ and χ are defined, our best strategy is to reconstruct $f^\dagger$ in the diagram:

$$\bar{f} \approx f^\dagger := \varphi \circ \mathcal{G}^\dagger \circ \psi$$

and to approximate the operator $\mathcal{G}^\dagger$ with the operator

$$\bar{\mathcal{G}} := \chi \circ \bar{f} \circ \phi\,.$$

$$\begin{tikzcd}
\mathcal{U} \arrow[rr, "\mathcal{G}^\dagger"] \arrow[d, shift left, "\phi\ \textbf{Measurement}"] & & \mathcal{V} \arrow[d, shift left, "\textbf{Measurement}\ \varphi"] \\
\mathbb{R}^m \arrow[u, shift left, "\textbf{Reconstruction}\ \psi"] \arrow[rr, "f^\dagger"] & & \mathbb{R}^n \arrow[u, shift left, "\textbf{Reconstruction}\ \chi"]
\end{tikzcd}$$

A simple kernel method for $f^\dagger$

Given a kernel S, we approximate $f^\dagger : \mathbb{R}^m \to \mathbb{R}^n$ via optimal recovery **independently component wise**:

$$\bar{f}_j := \arg\min_{h \in \mathcal{H}_S} \|h\|_S \quad \text{s.t.} \qquad h(\phi(u_i)) = (\varphi(v_i))_j \qquad\qquad \text{for} \quad i = 1, \ldots, N.$$

which also has closed form solution given by kernel regression:

$$\bar{f}_j(\boldsymbol{u}) = S(\boldsymbol{u}, U)S(U, U)^{-1}\boldsymbol{v}_j.$$

where $U_i := \phi(u_i)$ and $V_i := \varphi(v_i)$.
This can be interpreted as recovering $f^\dagger$ with a matrix valued kernel with diagonal entries (beyond this talk).

Why such a simple method?

The kernel S can be a standard kernel such as the linear[5], squared exponential or Matérn kernel. This simple choice already offers several advantages:

1. Low cost in training (< 5 seconds on a workstation) and at inference (in the low-medium data regime).
2. Competitive accuracy.
3. Empirically robust to choice of hyper-parameters/kernels.
4. Simple to implement: several libraries solve this problem out of the box.
5. The Gaussian process interpretation provides uncertainty quantification.
6. Convergence guarantees (beyond this talk).

[5] Equivalent to doing linear regression

Table of Contents

Experimental protocol

We compare the test performance of our method using the examples from two comparison papers[6,7] and the best-reported test relative L^2 loss.

	Low-data regime			High-data regime			
	Burger's	**Darcy problem**	**Advection I**	**Advection II**	**Hemholtz**	**Structural Mechanics**	**Navier Stokes**
DeepONet	2.15%	2.91%	0.66%	15.24%	5.88%	5.20%	3.63%
POD-DeepONet	1.94%	2.32%	0.04%	n/a	n/a	n/a	n/a
FNO	1.93%	2.41%	0.22%	13.49%	1.86%	4.76%	0.26%
PCA-Net	n/a	n/a	n/a	12.53%	2.13%	4.67%	2.65%
PARA-Net	n/a	n/a	n/a	16.64%	12.54%	4.55%	4.09%
Linear	36.24%	6.74%	2.15×10^{-13}%	11.28%	10.59%	27.11%	5.41%
Kernel method	2.15%	2.75%	2.75×10^{-3}%	11.44%	1.01%	5.18%	0.12%

Table: Summary of numerical results. When methods in their original work present variation, we report the best accuracy.

[6] Maarten V. de Hoop et al. *The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks*. 2022.

[7] Lu Lu et al. "A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data". In: *Computer Methods in Applied Mechanics and Engineering* 393 (2022).

Inverse problem for Darcy's flow

Let $D = (0,1)^2$ and consider the two-dimensional Darcy flow problem[8]:

$$
\begin{aligned}
-\nabla \cdot (u(x)\nabla v(x)) &= f, \quad x \in D \\
u(x) &= 0, \quad \partial D
\end{aligned}
$$

In this case, we are interested in learning the mapping from the permeability field u to the solution v (here f is considered fixed):

$$\mathcal{G}^\dagger : u(x) \mapsto v(x).$$

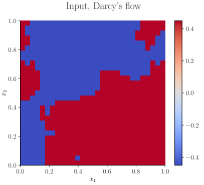
The coefficient u is sampled by $u = \psi(\mu)$ where $\mu = \mathcal{GP}(0, (-\Delta + 9I)^{-2})$ is a Gaussian random field and ψ is binary function.

[8] Lu et al., "A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data".

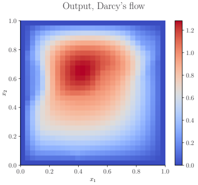
Low-data regime: Inverse problem for Darcy's flow

Method	Accuracy
DeepONet	2.91 %
FNO	2.41 %
POD-DeepONet	2.32 %
Linear Regression	6.74 %
GP (Matérn kernel)	2.75%

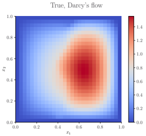
Table: L^2 relative error on the Darcy problem.



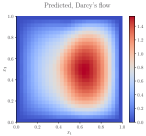
(a) Input



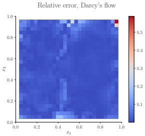
(b) Output



(c) True



(d) Predicted



(e) Relative Error

High-data regime: Navier-Stokes

In the periodic domain $\mathcal{D} = [0, 2\pi]^2$, the vorticity-stream $(\omega - \psi)$ formulation of the incompressible Navier-Stokes equations is

$$\begin{aligned} \frac{\partial w}{\partial t} + (v \cdot \nabla)\omega - \nu \Delta \omega &= f \\ \omega &= -\Delta \psi \\ \int_D \psi &= 0 \\ v &= \left(\frac{\partial \psi}{\partial x_2}, -\frac{\partial \psi}{\partial x_1} \right) \end{aligned}$$

The map of interest is the map from the forcing term f to the vorticity field w at a given time $t = T$:

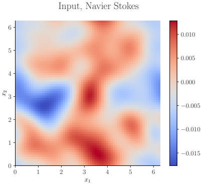
$$\mathcal{G}^\dagger : f \mapsto w(\cdot, T).$$

The forcing is sampled from a centered Gaussian field, $f \sim \mathcal{GP}(0, (-\Delta + 3^2 I)^{-4})$.

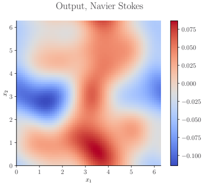
High data regime: Navier-Stokes

Method	Accuracy
DeepONet	3.63 %
FNO	0.26 %
PCA-Net	2.32 %
Linear Regression	5.41 %
GP (Matérn kernel)	0.12%

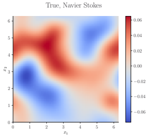
Table: L^2 relative error on Navier-Stokes.



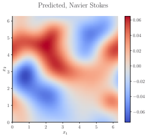
(a) Input



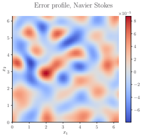
(b) Output



(c) True



(d) Predicted



(e) Error

Two versions of the advection problem

Let $D = (0,1)$ and consider the one-dimensional wave advection equation:

$$\frac{\partial v}{\partial t} + \frac{\partial v}{\partial x} = 0 \quad x \in (0,1), t \in (0,1]$$
$$v(x,0) = u_0(x) \quad x \in (0,1)$$

with periodic boundary conditions. We learn the operator mapping the initial condition to the solution at time $t = 0.5$:

$$\mathcal{G} : u_0(x) \mapsto v(x, 0.5).$$

The two versions differ in their initial conditions[9],[10]:

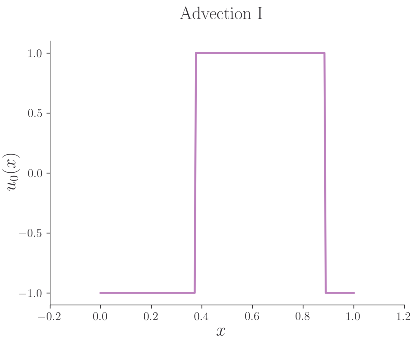
$$u_0(x) = h\mathbf{1}_{\{c-\frac{w}{2}, c+\frac{w}{2}\}} \qquad (c, w, h) \sim \mathcal{U} \qquad \text{(Advection I)}$$
$$u_0(x) = -1 + 2\mathbf{1}\{\tilde{u}_0 \geq 0\} \qquad \tilde{u}_0 \sim \mathcal{GP}(0, (-\Delta + 3^2)^{-2}) \qquad \text{(Advection II)}$$

---

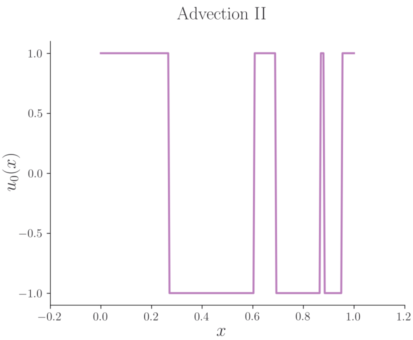
[9] Lu et al., "A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data".

[10] Hoop et al., *The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks*.

Two versions of the advection problem



(a) Advection I: initial condition



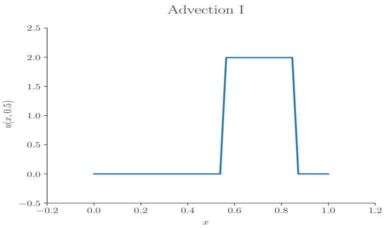
(b) Advection II: initial condition

Figure: The two versions of the advection problem

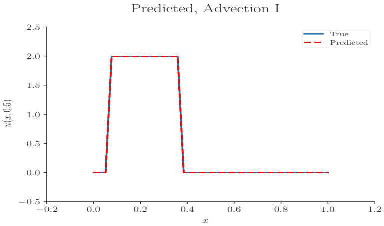
Advection I

Method	Accuracy
DeepONet	0.66 %
FNO	0.22 %
POD-DeepONet	0.04 %
Linear Regression	2.15×10^{-13}%
GP (Matérn kernel)	2.75×10^{-3}%

Table: L^2 relative error for the advection I.



(a) Input

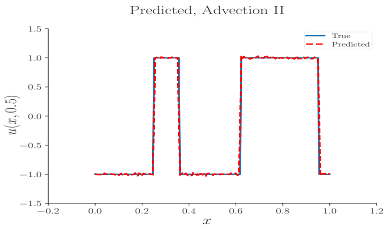


(b) Prediction by Linear regression

Advection II

Method	Accuracy
FNO	13.49%
DeepONet	15.24%
PCA-Net	12.53%
Linear Regression	11.28%
GP (Matérn kernel)	11.44%

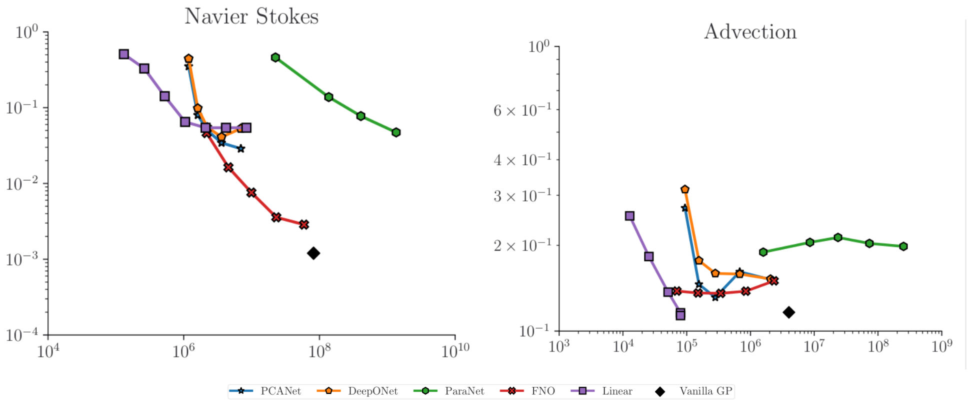
Table: L^2 relative error for advection II.



(a) Prediction by Linear regression

Inference complexity: high data regime

In the "high data" regime (10000 points), vanilla kernel method achieves high accuracy at the cost of complexity.



Data taken from Hoop et al., *The Cost-Accuracy Trade-Off In Operator Learning With Neural Networks*.

Conclusion

Our key contributions are:

- A simple, low-cost, and competitive kernel method for operator learning, which is a good baseline for many tasks.
- Convergence guarantees for this method.

Going beyond simple kernel methods:

- More complex matrix-valued kernels (non-diagonal, hierarchical kernels).
- "Non-vanilla" kernel methods: random Fourier features, inducing points ...

Paper coming out next week!