Lecture 5: Neural Operators 1, Background

Chris Budd[1]

[1]University of Bath

Seattle, June 2025

Some papers/books to look at

- Courant and Hilbert *Methods of mathematical physics Volumes 1,2*
- Stuart et. al. *Fourier Neural Operator for PDEs*
- Kovachi et. al. *Operator learning: algorithms and analysis*
- Boullé and Townsend *Learning elliptic PDEs*
- Halko et. al. *Finding structure with randomness*

Motivation: Solution Operators

Have studied using PINNS to solve PDE problems of the form

$$u_t = F(\mathbf{x}, u, \nabla u, \nabla^2 u) \quad \text{with BC},$$

$$u(0, x) = u_0(x)$$

At time T we have the solution $u_T(x) \equiv u(x, T)$.

Solution $u(x, t)$ for all x and t is obtained by minimising a function directly associated with the PDE eg. residual.

Neural Operator methods take a different approach

- Consider u_T as a function F of u_0. $u_T = F(u_0)$
- F is an operator mapping one infinite dimensional function space to another $F : A \to B$. eg. $A, B = H^1(\Omega)$
- Train a Neural Operator NN to approximate this operator note infinite dimensions
- Train it by generating a (large) set of solution pairs (u_0^i, u_T^i)

Can generate solution pairs using a (conventional) numerical method eg. Finite Element, Pseudo-Spectral, Symplectic.

eg. ERA5 data for 24 hour weather forecasts.

Example 0: A finite dimensional problem [Halko et. al.]

- Have an $n \times n$ matrix A
- Have a random set of N vectors $\mathbf{x}_i$
- Compute the N matrix vector products

$$A\,\mathbf{x}_i = \mathbf{y}_i$$

Question Construct the matrix A from the set of N solution pairs $(\mathbf{x}_i, \mathbf{y}_i)$

Methodology Use the (radomized) SVD to contruct an orthogonal basis for the range space of A spanned by the vectors y_i

Example 1: A linear ODE system

Consider the linear ODE

$$\frac{d\mathbf{u}}{dt} = A\,\mathbf{u}, \quad \mathbf{u}(0) = \mathbf{u}_0, \quad \mathbf{u} \in R^n.$$

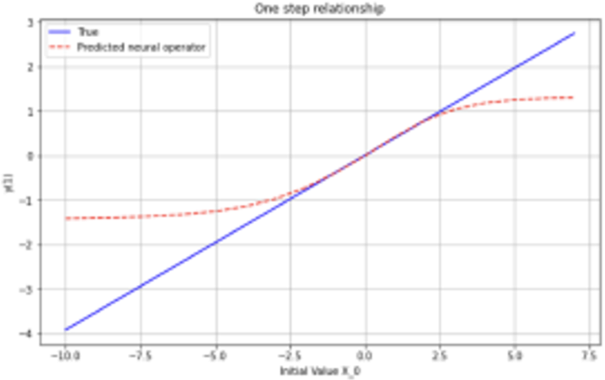
Solution

$$\mathbf{u}(T) = e^{A\,T}\mathbf{u}_0 \equiv B\,\mathbf{u}_0$$

$$B \equiv e^{AT} = I + AT + \frac{A^2T^2}{2!} + \frac{A^3T^3}{3!} + \ldots$$

Properties of the solution operator

- Operator is linear
- Operator is continuous over any subset of R^n
- Can easily learn the matrix B from data pairs if we assume that the operator is linear in advance!
- If we learn B from a subset of the data pairs then we can extrapolate this to ALL data pairs
- This is NOT true if don't make the linearity assumption. Many NN methods will locally approximate the operator to be linear, but will not give this as a global approximation.


One step relationship
True
Predicted neural operator
y(1)
Initial Value X_0
3
2
1
0
−1
−2
−3
−4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

Latent space description of the operator

Let A have eigenvectors ϕ_i so that

$$A\,\phi_i = \lambda_i \phi_i$$

Set $\mathbf{u} = \sum a_i(t)\,\phi_i$ then

$$\frac{d\mathbf{u}}{dt} = \sum \frac{da_i}{dt}\phi_i = \sum Au = \sum \lambda_i a_i \phi_i$$

so that

$$\frac{da_i}{dt} = \lambda_i a_i \implies a_i = a_i(0)e^{\lambda_i t}$$

Assume A is symmetric. Then can set

$$\phi_i^T \phi_j = \delta_{ij}$$

Hence

$$\mathbf{u}(T) = \sum \phi_i^T \mathbf{u}(0) e^{\lambda_i T} \phi_i.$$

Takes the form of

- **Encoder**: $\phi_i^T \mathbf{u}_0$.
- **Latent space evolution**: $e^{\lambda_i T}$
- **Decoder** Multiply by ϕ_i

We will mimic this structure in the design of Neural Operators eg. **Deep-O-Net**

Example 2: Parabolic PDEs

Consider the parabolic PDE [picture]

$$u_t = u_{xx} + f(x), \quad x \in [0, 2\pi], \quad u(0, x) = u_0(x), \quad periodicBC$$

We can express $u(x, t)$ in terms of convolutional integral operators:

$$u(x, t) = G * u_0 + H * f \equiv \int_0^{2\pi} G(x-y, t) u_0(y)\, dy + \int_0^{2\pi} H(x-y, t) f(y)\, dy$$

These operators act on the infinite dimensional space $L^2[0, 2\pi]$.

Can find $G(z, t)$ and $H(z, t)$ **explicitly** using a Fourier series.

This construction will motivate the construction, and use,of the FNO (Fourier Neural Operator) in the next Lecture

Fourier Series

As u and f are 2π periodic we can set:

$$u(x,t) = \sum_j c_j(t) e^{ijx}, \quad f(x) = \sum_j f_j e^{ijx},$$

Substituting into the PDE we have

$$\frac{du_j}{dt} = -j^2 u_j(0) + f_j,$$

with

Hence

$$c_j(T) = e^{-j^2 T}\left(c_j(0) - \frac{f_j}{j^2}\right) + \frac{f_j}{j^2}, \quad c_0(T) = c_0(0) + f_0 T$$

with

$$c_j(0) = \frac{1}{2\pi}\int_0^{2\pi} e^{-ijy} u_0(y)\, dy, \quad f_j = \frac{1}{2\pi}\int_0^{2\pi} e^{-ijy} f(y)\, dy.$$

Hence

$$u(x,T) = \int_0^{2\pi} \frac{1}{2\pi} \sum_j e^{ij(x-y)} e^{-j^2 T} u_0(y)\, dy$$

$$+ \int_0^{2\pi} \frac{1}{2\pi} \sum_j e^{ij(x-y)} j^{-2} f(y)\, dy + \dots$$

So we can see that this has the correct integral form with

$$G(z,T) = \sum_j \frac{1}{2\pi} e^{-j^2 T} e^{ijz}, \quad H(z,t) = \sum_j \frac{1}{2\pi} j^{-2} e^{ijz} + \dots$$

Trivially $G(z,T)$ has Fourier Coefficents

$$G_j = \frac{1}{2\pi} e^{-j^2 T}.$$

Learning G and H

- Suppose for a fixed $f(x)$ we have lots of solution pairs $(u_0^k(x), u_T^k(x))$ ($k = 1..N$ random set)
- Use FFT to find the Fourier coefficients $u_0^k \to u_0^{k,j}, u_T^k \to u_T^{k,j}, f \to f_j$
- For each j find the FCs of G and H by solving the minimisation problem

$$(G_j, H_j) = \text{argmin}_k \| G_j u_0^{k,j} + H_j f_j - u_T^{k,j} \|$$

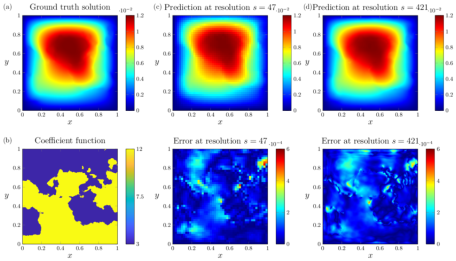
This motivates the construction of the Fourier Neural Operator (FNO) in the next lecture.

Example 3: Darcy Problem

The Darcy problem relates a permeability $a(x)$ to a velocity field $u(x)$

$$-\nabla \cdot (a(x)\nabla u) = f(x) \quad x \in \Omega, \quad u = 0 \quad x \in \partial\Omega.$$

This induces a (nonlinear) map $N : a \to u, \quad N : L^2(\Omega) \to H_0^1(\Omega)$



We can approximate this map using the Finite Element Method

$$u(x) \approx U(x) = \sum U_i \, \phi_i(x).$$

$$-\nabla \cdot (a(x)\nabla u) = f \implies \int a(x)\nabla u(x) \cdot \nabla \phi_i(x) \, dx = \int f(x)\phi_i(x) \, dx \equiv f_i$$

Giving the linear system

$$A\mathbf{U} = \mathbf{f}, \quad \mathbf{U}_i = U_i, \quad \mathbf{f}_i = f_i, \quad A_{ij} = \int a(x) \, \nabla \phi_i \cdot \nabla \phi_j \, dx.$$

Hence we can approximate the nonlinear map via:

$$\mathbf{U} = A^{-1} \, \mathbf{f}$$

And ...

We can LEARN this map by

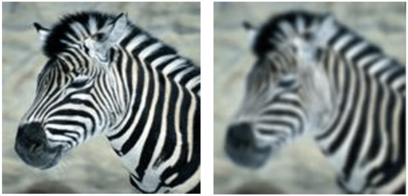
- Doing lots of finite element calculations to find solution pairs $(a(x), u(x))$
- Learn the operator between these pairs (see next lecture).

Pictures

In image processing a picture is often thought of as a high dimensional vector $\mathbf{z} \in R^n$.
Can also think of it as a function $f(x,y)$ $f : R^2 \to R$. Image processing is then an operation on an infinite dimensional function space.

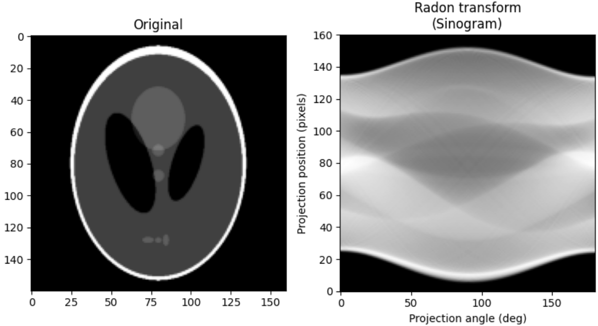
Example 1: Blurring

$$f \to G * f(x,y) = \int\int G(x-x', y-y')\, f(x',y')\, dx'dy'$$



Example 2: Radon Transform in Tomography

$$f(x,y) \to Rf(\theta,d) = \int f((z\sin(\theta) + d\cos(\theta)), (-z\cos(\theta) + d\sin(\theta))\, dz$$



Methods such as FNO work as much as possible in the infinite-dimensional function space.

They Construct and train Neural Operators which are approximations to the true operator (or its inverse) which are independent of the resolution of the underlying function/image

Nonlinear problems and a warning

Consider now the nonlinear parabolic PDE

$$u_t = u_{xx} + f(x,u), \quad u(0) = u(1) \quad u(0,x) = u_0(x)$$

This does not always induce a continuous map from $u(0,x) \to u(1,x)$.

- If $f(x,u)$ is Globally Lipshitz in x and u then all is OK
- If not then we may have problems
- See Case Study One!

Example

Let

$$f(x, u) = u^2, \quad u_0(x) = \gamma > 0$$

Then

$$u(1, x) = \frac{\gamma}{1-\gamma}.$$

Map is only continuous on the interval $\gamma \in [0, 1)$

If we train only on data with $\gamma < 1$ we will get a false result if we try to extend to $\gamma > 1$.

Case Study: Learning the Green's function of a linear elliptic operator

[Boullé + Townsend: https://arxiv.org/pdf/2102.00491]

Have an elliptic PDE

$$-\nabla \cdot (a(x)\nabla u) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset R^3, \quad u = 0, \quad \mathbf{x} \in \partial\Omega$$

There exists a Green's function $G(\mathbf{x}, \mathbf{y})$ so that

$$u(\mathbf{x}) = G * f \equiv \int_\Omega G(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) \, d\mathbf{y}.$$

$$G : C^\alpha \to C^{2,\alpha} \quad \text{Hölder Spaces}$$

$G(x, y)$ is a compact Hilbert-Schmidt operator, closely approximated by a finite dimensional operator $G_N(x, y)$

$$G(x, y) = \sum_{i=0}^{\infty} \frac{\phi_i(x)\ \phi_i(y)}{\lambda_i} = \sum_{i=0}^{N} \frac{\phi_i(x)\ \phi_i(y)}{\lambda_i} + \mathcal{O}\left(e^{-N^{1/4}}\right),$$

$$G_N(x, y) = \sum_{i=0}^{N} \frac{\phi_i(x)\phi_i(y)}{\lambda_i}$$

Aim: Assuming that $a(x)$ is unknown, to learn $G_N(x, y)$ and hence closely approximate $G(x, y)$

Methodology Learn $G_N(x, y)$ from a set of $M \approx N$ solution pairs

$$\mathcal{M} = \{(f_j(x), u_j(x)), j = 1 \ldots M\}$$

Draw $f_i(x)$ from a random Gaussian-Process of functions on Ω and use a randomised SVD based approach [see HalkoSVD] to construct G_N

Theorem [Boullé+Townsend] If $M = \epsilon^{-6} \log^4(1/\epsilon)$ then the approximation to G converges almost surely, with error E_ϵ so that

$$E_\epsilon < \Gamma_\epsilon^{-1/2} \epsilon \log^3(1/\epsilon), \quad \text{i.e.} \quad E \approx \mathcal{O}(M^{-1/6})$$

Γ_ϵ is a measure of the quality of the training data