Preliminary Report - Neural Operator

Nadav Port

Advisor: Professor Colin Cotter

August 1, 2024

Contents

1 Introduction

Machine learning, and in particular Deep Neural Networks (DNNs), have empirically proven themselves in diverse applications, such as image classification, natural language processing, text-to-speech, and more. Most of these problems can be formulated as approximating maps between finite-dimensional spaces, however big those spaces may be, e.g. space of RGB pixel values to labels. Nevertheless, many problems require a more general, infinite-dimensional setting. These are concerned with operators, which are maps between infinite-dimensional spaces, such as the abundance of PDEs lacking analytical solutions. These gave rise to machine learning methods for approximating operators, and has been termed ”operator learning”.

A few DNN architectures and theories have been proposed for operator learning, first appearing in 1995 [2]. More recent methodologies include Multiwavelet networks [3], DeepONetworks [4], and Neural Operators [7].

This work will follow the latter approach, first proposed as Graph and Fourier Neural Operators [5] [6], and developed further in [1] [7]. We aim to describe this construction under the extensively researched finite element method, taking advantage of its theory and software. We use the nodal basis of finite element spaces to represent input and output functions to the Neural Operator, and thus provide an explicit formula for the calculations that compose the Neural Operator

Ultimately, we have two broad goals. The first is to replicate known results of Neural Operators under our finite element framework. Secondly, we wish to discover relations between the architecture of Neural Operators, i.e. its hyperparameters, and how well they approximate a given problem. Hopefully, one could use these relations to one’s benefit while attempting to approximate solutions to a complicated PDE.

2 Literature Review

2.1 Machine learning framework

The Neural Operator is, first and foremost, a data-driven approximation that lies in the realm of machine learning. Thus, we need to consider a supervised learning approach for operators in infinite-dimensions.

Given an operator between two infinite-dimensional function spaces, $\mathcal{G} : \mathcal{A} \to \mathcal{U}$, we'd like to construct a surrogate operator $\mathcal{N}$ that will "learn" to imitate the action of $\mathcal{G}$ to arbitrary precision. Let $\{(a^{(i)}, u^{(i)})\}_{i=1}^{S} \subset \mathcal{A} \times \mathcal{U}$ be pairs of input-outputs, i.e. $u(x) = (\mathcal{G}a)(x)$, where $a^{(i)} \sim \mu$ are i.i.d samples drawn from a probability measure on $\mathcal{A}$. The approximating operator $\mathcal{N}_\theta : \mathcal{A} \to \mathcal{U}$ (later to be introduced as the Neural Operator) is an infinite-dimensional operator that is dependent on parameters $\theta \in \mathbb{R}^p$ for some $p \in \mathbb{N}$. We wish to minimize w.r.t. θ the approximation error, also known as true risk,

$$\mathbb{E}_{a\sim\mu} \|(\mathcal{G}a)(x) - (\mathcal{N}_\theta a)(x)\|_{\mathcal{U}}. \tag{2.1}$$

In general, we don't fully know the distribution μ, we only have access to a certain number of samples. Thus, as is customary in machine learning [8] (ch. 8), we turn to empirical-risk minimization,

$$\min_{\theta\in\mathbb{R}^p} \frac{1}{S} \sum_{i=1}^{S} \|u^{(i)}(x) - (\mathcal{N}_\theta a^{(i)})(x)\|_{\mathcal{U}}, \tag{2.2}$$

which approximates minimization of the true risk.

We remark that this discussion is still in an infinite-dimensional setting, as $\mathcal{N}_\theta$ is not discretized.

2.2 Neural Operators

Let $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ be a nonlinear operator acting between Banach function spaces. We denote the input and output functions as $a : \Omega \subset \mathbb{R}^k \to \mathbb{R}^{d_a}$, $u : \Omega' \subset \mathbb{R}^{k'} \to \mathbb{R}^{d_u}$ respectively.

Definition 2.1 — Neural Operator. A Neural Operator $\mathcal{N}$ is constructed as follows [1],

$$\begin{aligned} &\mathcal{N} : \mathcal{A} \to \mathcal{U} \\ &\mathcal{N} \overset{\text{def.}}{=} \mathcal{Q} \circ \mathcal{L}_L \circ \ldots \circ \mathcal{L}_1 \circ \mathcal{R} \end{aligned} \tag{2.3}$$

where

- **Lifting operator** - $\mathcal{R}$ is a pointwise operator that maps the input function a to a D-valued function, where $D \geq d_a$.
- **Lowering operator** - $\mathcal{Q}$ (a.k.a "projection" operator) is a pointwise operator that maps the output from $\mathcal{L}_L$, which is a D-valued function, to the output in $\mathcal{U}$.

- **Inner-layer Operator** - $\{\mathcal{L}_l\}_{l=1}^L$ are non-local, non-linear operators defined by,

$$(\mathcal{L}_l \boldsymbol{v})(x) \overset{\text{def.}}{=} \sigma\left(W_l \boldsymbol{v}(x) + \boldsymbol{b}_l + \sum_{m=1}^{M} \langle T_{l,m}\boldsymbol{v}(x), \psi_m(x)\rangle \psi(x) \right), \tag{2.4}$$

where σ is a nonlinear pointwise function, $T_{l,m}, W_l$, and $\boldsymbol{b}_l$ are matrices and vector of *learnable* network parameters of dimensions $D \times D$, $D \times 1$, $\{\psi\}_{m=1}^M$ are called projection functions, and the inner product $\langle \cdot, \cdot \rangle$ is the L^2 inner product.

Remark: The Neural Operator $\mathcal{N}$ is dependent on the learnable parameters in $T_{l_m}, W_l, \boldsymbol{b}_l$, and those appearing in $\mathcal{R}, \mathcal{Q}$. We concatenate these parameters in one vector $\theta \in \mathbb{R}^p$, and we'd sometimes write $\mathcal{N}_\theta$ instead of $\mathcal{N}$ to emphasize the parameter dependence.

2.3 Neural Operator Theory

The theory of Neural Operators is still mostly unestablished, but a few results regarding approximation theory had been proven. We will state a simplified version of the one most important to us, and reference the relevant papers for a proof. Generally, we assume the function domains Ω to be bounded with Lipschitz boundary, and we'll restrict ourselves to Sobolev Hilbert spaces $H^k(\Omega) = W^{k,2}(\Omega)$.

Theorem 2.2 — D convergence. Let $\Omega \subset \mathbb{R}^n$, $\mathcal{G} : H^s(\Omega, \mathbb{R}^k) \to H^{s'}(\Omega, \mathbb{R}^{k'})$ a continuous operator acting between Sobolev spaces, where $s, s' \in \mathbb{N}_0$, $k, k' \in \mathbb{N}$. Then, for any fixed $L \geq 1$, $M \geq 0$, and $\forall \epsilon > 0$ there exists a Neural Operator $\mathcal{N}_\theta$ for some $\theta \in \mathbb{R}^p$, such that

$$\forall a \in K \quad \|(\mathcal{G}a)(x) - (\mathcal{N}_\theta a)(x)\|_{H^{s'}} \leq \epsilon, \tag{2.5}$$

where $K \subset \mathcal{A}$ is a compact set. Furthermore, we can choose the functions ψ_m in (2.4) to be the identity, or more sophisticated functions.

L, M are fixed in the above theorem, (ψ_m) can be set to unity, and we assume θ is optimized, then the only possible variability in the Neural Operator architecture is in the number of lifting channels D, meaning $\|(\mathcal{G}a)(x) - (\mathcal{N}_{\theta(D)}a)(x)\|_{H^{s'}} \xrightarrow{D\to\infty} 0$. The full statement and proof are theorem 1.2 and appendix A.6 in [1].

Although Theorem 2.2 promises that in theory the error between the surrogate and original operators converges to zero, in practicality it raises questions. For example, what's the rate of decline of the error? For a given D, which choice of hyperparameters, $L, M, (\psi_m)$ provides the least error? Under limited computational resources (time and/or memory), what combination of hyperparameters is the best? If the total number of parameters p is to remain constant, should we increase D on behalf of L, or vice versa?

Confined to specific PDE examples, we hope to numerically analyze some of these questions in the final report.

References

[1] Lanthaler, S., Li, Z., and Stuart, A. M. (2023). The Nonlocal Neural Operator: Universal Approximation (Version 1). arXiv. http://doi.org/10.48550/ARXIV.2304.13221

[2] Chen, T., & Chen, H. (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, *6*(4), 911–917. https://doi.org/10.1109/72.392253

[3] Gupta, G., Xiao, X., & Bogdan, P. (2021). Multiwavelet-based Operator Learning for Differential Equations (Version 2). arXiv. http://doi.org/10.48550/ARXIV.2109.13459

[4] Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021, March 18). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*. Springer Science and Business Media LLC. http://doi.org/10.1038/s42256-021-00302-5

[5] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Neural Operator: Graph Kernel Network for Partial Differential Equations (Version 1). arXiv. http://doi.org/10.48550/ARXIV.2003.03485

[6] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier Neural Operator for Parametric Partial Differential Equations (Version 3). arXiv. http://doi.org/10.48550/ARXIV.2010.08895

[7] Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Neural Operator: Learning Maps Between Function Spaces (Version 6). Arxiv. http://doi.org/10.48550/ARXIV.2108.08481

[8] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. MIT press.