Title: Neural operators

URL: https://en.wikipedia.org/wiki/Neural_operators

PageID: 74636983

Categories: Category:Deep learning

Neural operators are a class of deep learning architectures designed to learn maps between infinite-dimensional function spaces . Neural operators represent an extension of traditional artificial neural networks , marking a departure from the typical focus on learning mappings between finite-dimensional Euclidean spaces or finite sets. Neural operators directly learn operators between function spaces; they can receive input functions, and the output function can be evaluated at any discretization. [1] [2]

The primary application of neural operators is in learning surrogate maps for the solution operators of partial differential equations (PDEs), [1] [2] which are critical tools in modeling the natural environment. [3] [4] Standard PDE solvers can be time-consuming and computationally intensive, especially for complex systems. Neural operators have demonstrated improved performance in solving PDEs [5] [6] compared to existing machine learning methodologies while being significantly faster than numerical solvers. [7] [8] [9] Neural operators have also been applied to various scientific and engineering disciplines such as turbulent flow modeling, computational mechanics, graph-structured data, [10] and the geosciences. [11] In particular, they have been applied to learning stress-strain fields in materials, classifying complex data like spatial transcriptomics, predicting multiphase flow in porous media, [12] and carbon dioxide migration simulations. Finally, the operator learning paradigm allows learning maps between function spaces, and is different from parallel ideas of learning maps from finite-dimensional spaces to function spaces, [13] [14] and subsumes these settings as special cases when limited to a fixed input resolution.

Operator learning

Understanding and mapping relationships between function spaces has many applications in engineering and the sciences. In particular, one can cast the problem of solving partial differential equations as identifying a map between function spaces, such as from an initial condition to a time-evolved state. In other PDEs this map takes an input coefficient function and outputs a solution function. Operator learning is a machine learning paradigm to learn solution operators mapping the input function to the output function.

Using traditional machine learning methods, addressing this problem would involve discretizing the infinite-dimensional input and output function spaces into finite-dimensional grids and applying standard learning models, such as neural networks. This approach reduces the operator learning to finite-dimensional function learning and has some limitations, such as generalizing to discretizations beyond the grid used in training.

The primary properties of neural operators that differentiate them from traditional neural networks is discretization invariance and discretization convergence. [1] Unlike conventional neural networks, which are fixed on the discretization of training data, neural operators can adapt to various discretizations without re-training. This property improves the robustness and applicability of neural operators in different scenarios, providing consistent performance across different resolutions and grids.

Definition and formulation

Architecturally, neural operators are similar to feed-forward neural networks in the sense that they are composed of alternating linear maps and non-linearities. Since neural operators act on and output functions, neural operators have been instead formulated as a sequence of alternating linear integral operators on function spaces and point-wise non-linearities. [1] Using an analogous

architecture to finite-dimensional neural networks, similar universal approximation theorems have been proven for neural operators. In particular, it has been shown that neural operators can approximate any continuous operator on a compact set. [1]

Neural operators seek to approximate some operator G : A → U {\displaystyle {\mathcal {G}}:{\mathcal {A}}\to {\mathcal {U}}} between function spaces A {\displaystyle {\mathcal {A}}} and U {\displaystyle {\mathcal {U}}} by building a parametric map G φ : A → U {\displaystyle {\mathcal {G}}_{\phi }:{\mathcal {A}}\to {\mathcal {U}}} . Such parametric maps G φ {\displaystyle {\mathcal {G}}_{\phi }} can generally be defined in the form

G φ := Q ■ σ (W T + K T + b T) ■ ■ ■ σ (W 1 + K 1 + b 1) ■ P , {\displaystyle {\mathcal {G}}_{\phi }:={\mathcal {Q}}\circ \sigma (W_{T}+{\mathcal {K}}_{T}+b_{T})\circ \cdots \circ \sigma (W_{1}+{\mathcal {K}}_{1}+b_{1})\circ {\mathcal {P}},}

where P , Q {\displaystyle {\mathcal {P}},{\mathcal {Q}}} are the lifting (lifting the codomain of the input function to a higher dimensional space) and projection (projecting the codomain of the intermediate function to the output dimension) operators, respectively. These operators act pointwise on functions and are typically parametrized as multilayer perceptrons . σ {\displaystyle \sigma } is a pointwise nonlinearity, such as a rectified linear unit (ReLU) , or a Gaussian error linear unit (GeLU) . Each layer t = 1 , … , T {\displaystyle t=1,\dots ,T} has a respective local operator W t {\displaystyle W_{t}} (usually parameterized by a pointwise neural network), a kernel integral operator K t {\displaystyle {\mathcal {K}}_{t}} , and a bias function b t {\displaystyle b_{t}} . Given some intermediate functional representation v t {\displaystyle v_{t}} with domain D {\displaystyle D} in the t {\displaystyle t} -th hidden layer, a kernel integral operator K φ {\displaystyle {\mathcal {K}}_{\phi }} is defined as

(K φ v t) (x) := ∫ D κ φ (x , y , v t (x) , v t (y)) v t (y) d y , {\displaystyle ({\mathcal {K}}_{\phi }v_{t})(x):=\int _{D}\kappa _{\phi }(x,y,v_{t}(x),v_{t}(y))v_{t}(y)dy,}

where the kernel κ φ {\displaystyle \kappa _{\phi }} is a learnable implicit neural network, parametrized by φ {\displaystyle \phi } .

In practice, one is often given the input function to the neural operator at a specific resolution. For instance, consider the setting where one is given the evaluation of v t {\displaystyle v_{t}} at n {\displaystyle n} points { y j } j n {\displaystyle \{y_{j}\}_{j}^{n}} . Borrowing from Nyström integral approximation methods such as Riemann sum integration and Gaussian quadrature , the above integral operation can be computed as follows:

∫ D κ φ (x , y , v t (x) , v t (y)) v t (y) d y ≈ ∑ j n κ φ (x , y j , v t (x) , v t (y j)) v t (y j) Δ y j , {\displaystyle \int _{D}\kappa _{\phi }(x,y,v_{t}(x),v_{t}(y))v_{t}(y)dy\approx \sum _{j}^{n}\kappa _{\phi }(x,y_{j},v_{t}(x),v_{t}(y_{j}))v_{t}(y_{j})\Delta _{y_{j}},}

where Δ y j {\displaystyle \Delta _{y_{j}}} is the sub-area volume or quadrature weight associated to the point y j {\displaystyle y_{j}} . Thus, a simplified layer can be computed as

v t + 1 (x) ≈ σ (∑ j n κ φ (x , y j , v t (x) , v t (y j)) v t (y j) Δ y j + W t (v t (y j)) + b t (x)) . {\displaystyle v_{t+1}(x)\approx \sigma \left(\sum _{j}^{n}\kappa _{\phi }(x,y_{j},v_{t}(x),v_{t}(y_{j}))v_{t}(y_{j})\Delta _{y_{j}}+W_{t}(v_{t}(y_{j}))+b_{t}(x)\right).}

The above approximation, along with parametrizing κ φ {\displaystyle \kappa _{\phi }} as an implicit neural network, results in the graph neural operator (GNO). [15]

There have been various parameterizations of neural operators for different applications. [7] [15] These typically differ in their parameterization of κ {\displaystyle \kappa } . The most popular instantiation is the Fourier neural operator (FNO). FNO takes κ φ (x , y , v t (x) , v t (y)) := κ φ (x − y) {\displaystyle \kappa _{\phi }(x,y,v_{t}(x),v_{t}(y)):=\kappa _{\phi }(x-y)} and by applying the convolution theorem , arrives at the following parameterization of the kernel integral operator:

(K φ v t) (x) = F − 1 (R φ ⋅ (F v t)) (x) , {\displaystyle ({\mathcal {K}}_{\phi }v_{t})(x)={\mathcal {F}}^{-1}(R_{\phi }\cdot ({\mathcal {F}}v_{t}))(x),}

where F {\displaystyle {\mathcal {F}}} represents the Fourier transform and R φ {\displaystyle R_{\phi }} represents the Fourier transform of some periodic function κ φ {\displaystyle \kappa _{\phi

}} . That is, FNO parameterizes the kernel integration directly in Fourier space, using a prescribed number of Fourier modes. When the grid at which the input function is presented is uniform, the Fourier transform can be approximated using the discrete Fourier transform (DFT) with frequencies below some specified threshold. The discrete Fourier transform can be computed using a fast Fourier transform (FFT) implementation.

Training

Training neural operators is similar to the training process for a traditional neural network. Neural operators are typically trained in some Lp norm or Sobolev norm . In particular, for a dataset { (a i , u i) } i = 1 N {\displaystyle \{(a_{i},u_{i})\}_{i=1}^{N}} of size N {\displaystyle N} , neural operators minimize (a discretization of)

L U ({ (a i , u i) } i = 1 N) := ∑ i = 1 N ■ u i – G θ (a i) ■ U 2 {\displaystyle {\mathcal {L}}_{\mathcal {U}}(\{(a_{i},u_{i})\}_{i=1}^{N}):=\sum _{i=1}^{N}\|u_{i}-{\mathcal {G}}_{\theta }(a_{i})\|_{\mathcal {U}}^{2}} ,

where ■ · ■ U {\displaystyle \|\cdot \|_{\mathcal {U}}} is a norm on the output function space U {\displaystyle {\mathcal {U}}} . Neural operators can be trained directly using backpropagation and gradient descent -based methods.

Another training paradigm is associated with physics-informed machine learning. In particular, physics-informed neural networks (PINNs) use complete physics laws to fit neural networks to solutions of PDEs. Extensions of this paradigm to operator learning are broadly called physics-informed neural operators (PINO), [16] where loss functions can include full physics equations or partial physical laws. As opposed to standard PINNs, the PINO paradigm incorporates a data loss (as defined above) in addition to the physics loss L P D E (a , G θ (a)) {\displaystyle {\mathcal {L}}_{PDE}(a,{\mathcal {G}}_{\theta }(a))} . The physics loss L P D E (a , G θ (a)) {\displaystyle {\mathcal {L}}_{PDE}(a,{\mathcal {G}}_{\theta }(a))} quantifies how much the predicted solution of G θ (a) {\displaystyle {\mathcal {G}}_{\theta }(a)} violates the PDEs equation for the input a {\displaystyle a} .

See also

Neural network

Physics-informed neural networks

Neural field

References

External links

neuralop – Python library of various neural operator architectures