Operator learning via Neural Network

Chunyan Li

Mar.1st 2022 - Mar.21th 2022

1 Introduction

In this note, we will briefly summarize main popular papers/ideas of combination of differential equations (DE, even integral equations) and Neural network. For the applications of neural network in differential equations fields, we could summarize them into four groups:

- Solve differential equations
  - For a given differential equation and solution data points of it, solve it using NN
- Learn parameter functions in differential equations
  - For a given DE with parameter function $a(x)$ and data set from DE, learn/infer/identify the parameter function $a(x)$
- Discover the specific dynamic systems
  - For data $\{x(t_i)\}$, assume there is a underling ODE/PDE system, but the format of this dynamic is unknown. Discover pde/ode from the data $\{x(t_i)\}$
- Operator learning / Discover the whole pde/ode systems
  - mappings between finite dimensional spaces
  - mappings between infinite dimensional spaces

2 Solve PDE/ODE system

Problem setup: For the given/known PDE or ODE system, one can solve pde/ode system by design the loss function via residual.
Method: The input of NN is (x, t), output is solution $u(x, t)$ for pde. The input of NN is (t), output is solution $z(t)$ for ode. Define the loss function as the residual of the known system (which is unsupervised learning, but you can play with part of supervised learning by adding a mse between data and predicted data in loss). The PINN embeds the differential equations into the loss of the 8 neural network using automatic differentiation. There are continuous model and timperal-discrete model. The derivative w.r.t spatial variable x are computed through AD (automatic differentiation). References: Rassiz (Brown university), Lulu (Uof Pennsylvania),

3 Learn/infer parameters in PDE/ODE system

Problem setup: One has a PDE/ODE system with unknown parameters p, and try to determine p using the data pairs (x, u). **Method:** Same idea with 1 but optimize the unknown parameters p in PDE/ODE system and the weights and bias of NN simultaneously. Reference: M. Raissi, Lulu

4 Discover dynamic system

Problem setup: Assume there is a underlying PDE/ODE system for the given dataset $\{x(t_i)\}$, but we don't know the format of this dynamic, and One use NN to learn/discover this underlying dynamic system. There are several types of questions: a. $dx/dt = f(x(t))$, b. $dx/dt = f(x, t)$, c.$dx/dt = f(x, t, \mu)$ where μ is a parameter independent of t. **Method:** represent the right-hand side f as a neural network, and combine with classical numerical time-stepping rules to define the loss function. While for the non-autonomous dynamic system or the parameterize dynamic system, you need to pay more attention on t and mu. Reference: M. Raissi

5 Operator learning

Operator learning: Problem setup: directly learn the mapping from the coefficient in PDE/ODE (initial conditions or boundary conditions) to the solution function. (Do we need to know the PDE/ODE?) **Method:** DeepOnet and Fourier neural operator and Graph neural operator etc.

Physics-informed neural network (PINN) proposed by Raissi [5, 6] set off a wave of using neural networks to solve partial differential equations (PDE). There are lots of works on solving partial differential equations and identifying parameters in dynamic systems and PDE using deep neural networks. But most of them focus on learning mappings between finite-dimensional Euclidean spaces. Recently, this has been generalized to operator learning that learning the mappings between infinite dimensional function spaces. In [1], the universal approximation of nonlinear operator is proposed in 1995 as follows:

Universal Approximation Theorem for Operator: Suppose that σ is a continuous non-polynomial function X is a Banach space, $K_1 \subset X$, $K_2 \subset R^d$ are two compact sets in x and R^d, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p, m constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in R, w_k \in R^d, x_j \in K_1, i = 1, ..., n, k = 1, ..., p, j = 1, ..., m$,such that

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma\left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k\right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk} \right| \leq \epsilon \tag{1}$$

holds for all $u \in V$ and $y \in K_2$.

Lu and his collaborator proposed DeepOnet [4] in 2019 based on this theorem by using two neural networks to represent branch part and trunk part. Basically, one can treat trunk-net is a set of function basis of function space $C(K_2)$ where $G[u](y)$ belongs. The branch-net is a set of functional basis of functional space defined on V where $u(x)$ belongs.

Graph kernel network [3] and Fourier neural operator [2] proposed by Stuart, Andrew & Anandkumar, Anima group in Caltech in 2020 rely on the main idea that the kernel integration can be viewed as an aggregation of messages as follows:

$$v_{i+1}(x) = \sigma\left(W v_i(x) + \underbrace{\int_D \kappa_\phi(x, y, a(x), a(y)) v_i(y) dy}_{\text{kernel integral}}\right) \tag{2}$$

where $i = 0, 1, ..., T - 1$.

This is a natural Res-Net structure in eq.2, so the main problem is how to implement the kernel integral term. If one learns this term in frequency domain using Fourier Transform, then, it could be represented by a linear layer and Fast Fourier Transform (FFT) could be used to accelerate computations.

5.1 Onsager Net

will be updated soon! The Ritz-net proposed by Weinan E [**Eritz**] start with variational problem instead of pde and directly use neural network to approximate/represent the minimizer of the interested variational problem and build the cost function using the variational principle.