AI4PDE

Kernels and Convolutions in Various Domains

Yunfeng Liao

19 July,2024

Presentation Overview

Sources

Literatures

- Neural Operator: Graph Kernel Network for Partial Differential Equations. Zongyi Li et al. **ICLR,2020**
- Multipole Graph Neural Operator for Parametric Partial Differential Equations. Zongyi Li et al. **CoRR,2020**
- FOURIER NEURAL OPERATOR FOR PARAMETRIC PDE. Zongyi Li et al. **ICLR,2021**
- Multiwavelet-based Operator Learning for Differential Equations. Gaurav Gupta, Xiongye Xiao et al. **NIPS,2021**
- COUPLED MULTIWAVELET NEURAL OPERATOR LEARNING FOR COUPLED PDE. Xiongye Xiao et al. **ICLR,2023**

Tutorials

- Partial Differential Equations. Evans.
- 微分方程数值解法. 李荣华,刘播.
- 数值分析. 李庆扬等.
- Finite Element Methods and Their Applications. Zhangxin Chen.
- Introductory Variational Calculus on Manifolds. Ivo Terek.
- From Fourier Analysis to Wavelets. Jonas Gomes, Luiz Velho.

Section Overview

Retrospect

Ways to Solve PDE

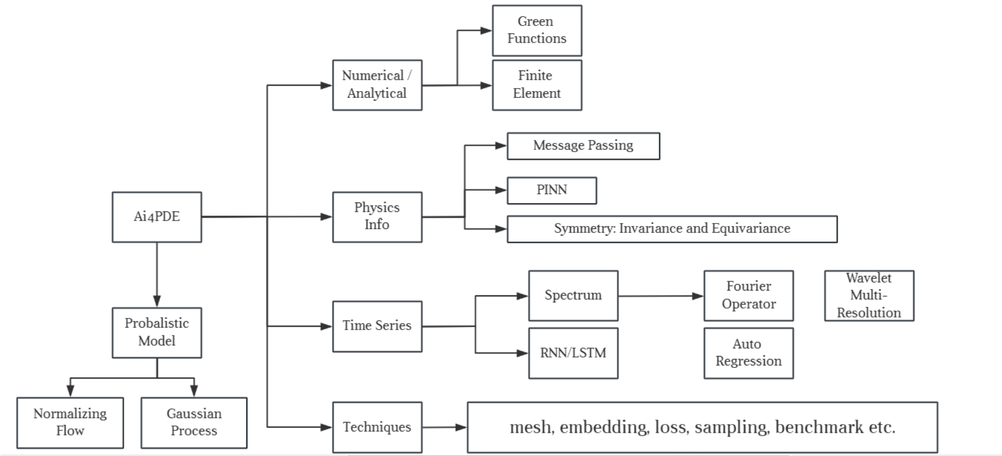


Fig. 1: Ways to Solve PDE with DL

Literatures

- (Review) Normalizing Flows for Probabilistic Modeling and Inference.
- Neural Ordinary Differential Equations. Ricky Chen. **2019**.
- PHYSICS-INFORMED KRIGING: A PHYSICS-INFORMED GAUSSIAN PROCESS REGRESSION METHOD FOR DATA-MODEL CONVERGENCE. XIU YANG. **2018**
- Physics-Informed CoKriging: A Gaussian-Process-Regression-Based Multifidelity Method for Data-Model Convergence XIU YANG. **2018**
- (Tutorial) C. E. Rasmussen C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006.

Why Bring in Probability

In the aforementioned scenario, a PDE problem is assumed to have accurate expressions, BC's, IC's and prior observations.
However, it is not the case in the reality. One can yield different observations o_i in the same location x due to systematic uncertainty. For instance, Gaussian Process methods assume $o_i(x) \sim N(\mu, \sigma^2(x))$.

Normalizing Flow

The spirit of normalizing flow is to transform a simple given distribution many times to obtain a desired distribution.

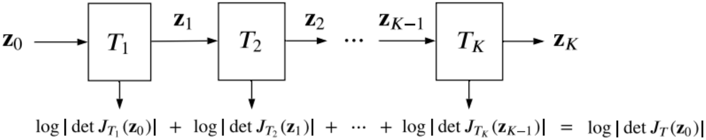


Fig. 2: Normalizing Flow

$$z_{n+1} = T_{n+1}(z_n), z_0 \sim p_0(z_0) \tag{1}$$

$$p_{n+1}(z_{n+1}) = p_n(T_n^{-1}(z_{n+1}))|\det J_{T_n^{-1}}(z_{n+1})| \tag{2}$$

$$p_{n+1}(z_{n+1}) = p_0(T_0^{-1}...T_n^{-1}(z_{n+1}))|\det J_{T_n^{-1}}(z_{n+1})|...|\det J_{T_0^{-1}}(z_1)| \tag{3}$$

Normalizing Flow

Let $\theta := \{\phi, \psi\}$ denote the learnable parameters of the transformations T_j and the original distribution $\mathbf{z}_0 \sim p_0$ respectively. It is common to use various divergences to measure the difference between two distributions and thus as a candidate of loss term.

The KL divergence is:

$$\begin{aligned} \mathscr{L}(\theta) &:= D_{KL}[p_{GT}(\mathbf{z_N})||p_N(z_N;\theta)] && (4) \\ &= -\mathbb{E}_{p_{GT}}[\ln p_N(z^N;\theta)] && (5) \\ &= -\mathbb{E}_{p_{GT}}[\ln p_0(T_0^{-1} \ldots T_n^{-1}(z_N);\psi,\phi)| + \sum_k \ln|\det J_{T_k^{-1}}(z_{k+1};\phi)|] && (6) \end{aligned}$$

where p_{GT} is short for the ground truth distribution. One can sample from p_{GT} and thereby apply gradient descent to update ϕ, ψ. **Interchange** p_{GT} **and** p_N, and get the so-called **backward KL divergence**, which can be evaluated by sampling from p_0.

Normalizing Flow

Some tricks can be used to ensure J_{T_n} is triangular so as to figure out $\det J_{T_n}$ more easily. A planar flow can be

$$\mathbf{z}' = \sigma(A\mathbf{z} + \mathbf{b}) \tag{7}$$

where A is triangular.

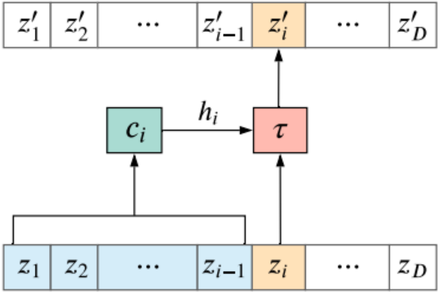


Fig. 3: Forward NF

NF in Solving ODE

An ODE can be evaluated by Euler method:

$$\mathbf{z}(t+h) = \mathbf{z}(t) + \int_t^{t+h} f(\mathbf{z}, s, \theta) ds \tag{8}$$

Theorem

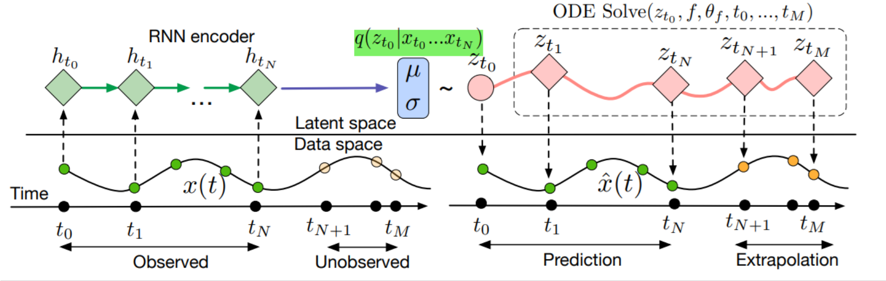
$\frac{d}{dt}\mathbf{z}(t) = f(\mathbf{z}(t), t)$ and f is L-continuous, then:

$$\frac{d}{dt} \ln p(\mathbf{z}(t)) = -\operatorname{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) \tag{9}$$

Thus we can define as below to evaluate $\mathbf{z}(t+h)$:

$$\frac{dz}{dt} = \text{NormalizingFlow}(\mathbf{z}(t)) \tag{10}$$

NF in Solving ODE



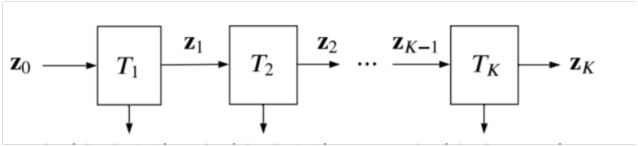


Fig. 4: NF solving

Section Overview

Spatial Convolution via Message Passing

Neural Operator

Neural Operator $\mathscr{F}$ is a map between two function spaces $\mathscr{F}: \mathscr{A} \to \mathscr{U}$ implemented by a neural network(NN) parametrized by Θ.

Example

$$\nabla \cdot (a(x)\nabla u(x)) = f(x), \forall x \in \Omega \tag{11}$$

$$u(x) = g(x), \forall x \in \partial\Omega \tag{12}$$

we wanna find a $\mathscr{F}$ s.t.

$$\mathscr{F}[a(x)] = u(x) \tag{13}$$

In fact, it can also expected that:

$$\mathscr{F}[f(x)] = u(x) \tag{14}$$

$$\mathscr{F}[g(x)] = u(x) \tag{15}$$

Motivation[1]

Motivation

$u(x)$ is more or less related to its neighbor $\{u(y) : d(x,y) < \delta\}$. Such relations often appear in the form of convolutions.

Example

Green functions $G(x,y)$. To solve $\Delta u(x) = f(x), x \in \Omega; u(x) = 0, x \in \partial\Omega$, $G(x,y)$ act as a kernel:

$$u(x) = \int_\Omega G(x,y) f(y) dy \tag{16}$$

Mean Value In Harmonic Fields If $\Delta u(x) = 0$, then:

$$u(\mathbf{x}_0) = \int_{\partial B(r)} u(\mathbf{x}_0 + \mathbf{n}r) dS \tag{17}$$

The point of view in GNN

A hidden layer is defined as:

$$h_{t+1}(x) = \sigma\left(Wh_t(x) + \int_{B(x,r)} \kappa_\phi(x, y, a(x), a(y))h_t(y)dy\right) \tag{18}$$

In the perspective of GNN, one can take all points within $B(x,r)$ as the neighbors of x, denoted by $N(x)$, and thereby construct a graph after discretization, i.e.

$$h_{t+1}(x) = \sigma\left(Wh_t(x) + \frac{1}{|N(x)|}\sum_{y \in N(x)} \kappa_\phi(e(x,y))h_t(y)dy\right) \tag{19}$$

where the edge weight $e(x,y)$ is determined by $x, y, a(x), a(y)$ jointly.

Discussion

Drawback

The scale of $N(x)$ is too large.

Nystrom approximation of the kernel. Use some randomly generalized sub-graphs instead.

Multi-scale Kernel[2]

Motivation

Kernels are needed to tell relations in different scales. Supposing K a discretized kernel, i.e., a matrix, it can be reached via low-rank factorization(via **SVD**):

$$K = K_1 + K_{12}K_2K_{21} + K_{12}K_{23}K_3K_{32}K_{21} + \ldots, \mathrm{rk}(K_1) > \mathrm{rk}(K_2) > \mathrm{rk}(K_3) > \ldots \quad (20)$$

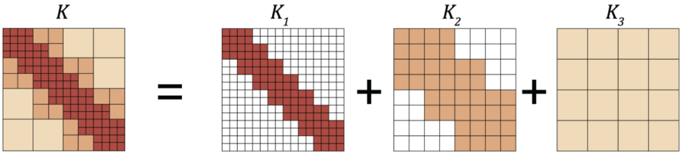


Figure 2: Hierarchical matrix decomposition

Fig. 5: Hierarchical matrix decomposition

Multi-scale Kernel

$v_l(x)$ denotes the feature field over different graphs. There 3 types of kernels $K_{l,l}, K_{l,l+1}, K_{l+1,l}$

$$K_{i,j} : v_i(x) \mapsto v_j(x) := \int \kappa_{\phi_{i,j}}(x, y, a(x), a(y)) v_i(y) dy \tag{21}$$

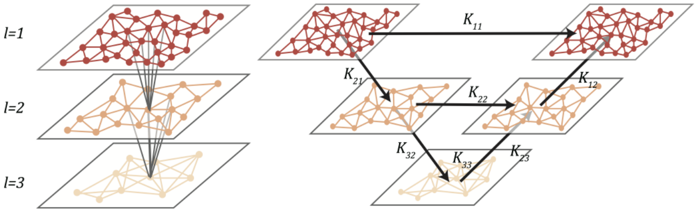


Figure 1: V-cycle

Fig. 6: V-Cycle

Architectures

The original is:

$$v^{t+1}(x) = \sigma(Wv^t + Kv^t) \tag{22}$$

The proposed **V-cycle algorithm** comes from the **low-rank decomposition**(20) and uses two feature fields v_D, v_U to **ensure updating chronologically**.

Downward Pass:

$$v_{D,l+1}^{t+1} = \sigma(v_{U,l+1}^{t} + K_{l+1,l} v_{D,l}^{t+1}) \tag{23}$$

Upward Pass:

$$v_{U,l+1}^{t+1} = \sigma((W_l + K_{l,l}) v_{D,l}^{t+1} + K_{l,l-1} v_{U,l-1}^{t+1}) \tag{24}$$

The entire structure is:

- Encode. $v_0 = P_\theta(x, a(x))$
- V-cycle. Downward and upward pass.
- Decode. $u = Q_\phi(v_T(x))$

Section Overview

Convolution in Frequency/Scale Domains

Fourier Transform :Convolution in Frequency Domain[3]

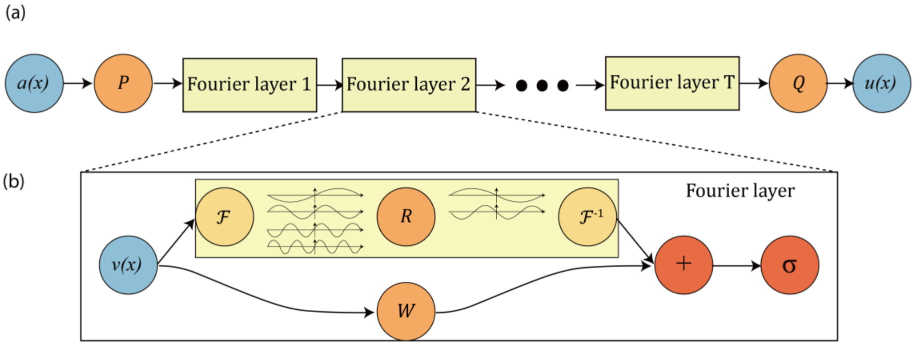


Fig. 7: Fourier Transform

Basics about Wavelets

Definition

Given a function ψ, termed **mother wavelet**, one can yield a family of wavelets via dilation and translation.

$$\psi_{a,b}(x) := \frac{1}{|a|^{\frac{1}{2}}}\psi(\frac{x-b}{a}) \tag{25}$$

Example

Harr wavelet.

$$\psi(x) := \mathbb{1}_{[0,\frac{1}{2})} - \mathbb{1}_{[\frac{1}{2},1]} \tag{26}$$

Basics about Wavelets[4]

Definition

$V_0 := \text{span}\{\phi_i\}$ s.t. $(\phi_i, \phi_j) = \delta_{ij}$. Let $\phi_i \equiv \phi_i^0$ We define V_k recursively: $V_{k+1} := \text{span}\{\phi_{j,l}^{k+1} := \sqrt{2}\phi_{j,2l}^{k}(\frac{x}{2})\}$. Therefore, $V_0 \subset V_1 \subset ... \subset V_{k+1}$. A decomposition tells $V_{k+1} = V_k \oplus V_k^{\perp} := V_k \oplus W_k$ This wavelet family is called **Dyadic wavelets**.

Now we focus on **orthogonal dyadic wavelets**. Denote the basis of V_k, W_k be $\phi_{j,l}^k, \psi_{j,l}^k$, we have: $(\phi_{j,l}^k, \psi_{m,n}^p) = \delta_{jm}\delta_{ln}\delta^{kp}$.
As what we do in Fourier series, a function $f \in L^2(R)$ can be decomposed as:

$$f(x) = \sum_{i,n,l} (f, \phi_{i,l}^n)\phi_{i,l}^n + (f, \psi_{i,l}^n)\psi_{i,l}^n \tag{27}$$

Two-Scale Relations: avoid computing piles of inner product

$$\mathbf{s}_l^n = [(f, \phi_{i,l}^n)]_{i=0}^{k-1}, \mathbf{d}_l^n = [(f, \psi_{i,l}^n)]_{i=0}^{k-1} \tag{28}$$

$$H_{ij}^{(0)} = \sqrt{2}\int \phi_i(x)\phi_j(2x)dx; H_{ij}^{(1)} = \sqrt{2}\int \phi_i(x)\phi_j(2x-1)dx \tag{29}$$

$$G_{ij}^{(0)} = \sqrt{2}\int \psi_i(x)\phi_j(2x)dx; G_{ij}^{(1)} = \sqrt{2}\int \psi_i(x)\phi_j(2x-1)dx \tag{30}$$

$$\Sigma_{ij}^{(0)} = \sqrt{2}\int \phi_i(2x)\phi_j(2x)dx; \Sigma_{ij}^{(1)} = \sqrt{2}\int \phi_i(2x-1)\phi_j(2x-1)dx \tag{31}$$

Decompose:

$$\mathbf{s}_{2l}^{n+1} = \Sigma^{(0)}(H^{(0)\,T}\mathbf{s}_l^n + G^{(0)\,T}\mathbf{d}_l^n); \mathbf{s}_{2l+1}^{n+1} = \Sigma^{(1)}(H^{(1)\,T}\mathbf{s}_l^n + G^{(1)\,T}\mathbf{d}_l^n) \tag{32}$$

Reconstruct:

$$\mathbf{s}_l^n = H^{(0)}\mathbf{s}_{2l}^{n+1} + H^{(1)}\mathbf{s}_{2l+1}^{n+1}; \mathbf{d}_l^n = G^{(0)}\mathbf{s}_{2l}^{n+1} + G^{(1)}\mathbf{s}_{2l+1}^{n+1} \tag{33}$$

Wavelet Analysis in PDE

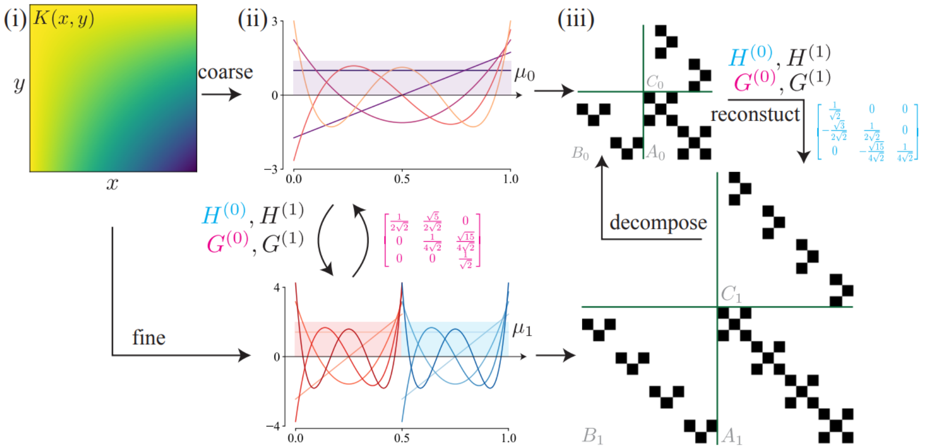


Fig. 8: Multiwavelet representation of the Kernel

Wavelet Neural Operator

T denotes the desired neural operator s.t. $T[a(x)] = u(x)$ and $P_j : L^2(\mathbb{R}) \to V_j$ is the projection operator. We can define a neural operator from V_j to itself by $T_n := P_n T P_n$ Suppose the lowest and highest resolution is L, n, one can write:

$$T_n = \sum_{i=L+1}^{n} (Q_i T Q_i + Q_i T P_{i-1} + P_{i-1} T Q_i) + P_L T P_L := \sum_{i=L+1}^{n} (A_i + B_i + C_i) + \overline{T} \quad (34)$$

where $Q_i := P_i - P_{i-1}$ is the projection operator onto W_i.
If s_l^n, d_l^n are the coefficients of $a(x)$, then the coefficients of $u(x)$ consists of three parts by (34):

$$U_{d,l}^n = A_n d_l^n + B_n s_l^n ; U_{s',l}^n = C_n d_l^n , U_{s,l}^L = \overline{T} s_l^L \quad (35)$$

Once we have these coefficients, we can use two-scale relation to reconstruct $u(x)$.

Wavelet Layer

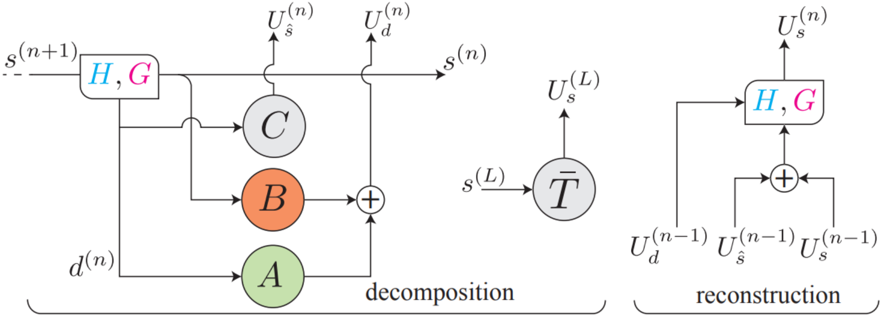


Fig. 9: Enter Caption

Coupled Wavelet layers in PDE[5]

Motivation

How to solve partial different equation**S**?

Example

$$\partial_t u = f(u, v) \tag{36}$$

$$\partial_t v = g(u, v) \tag{37}$$

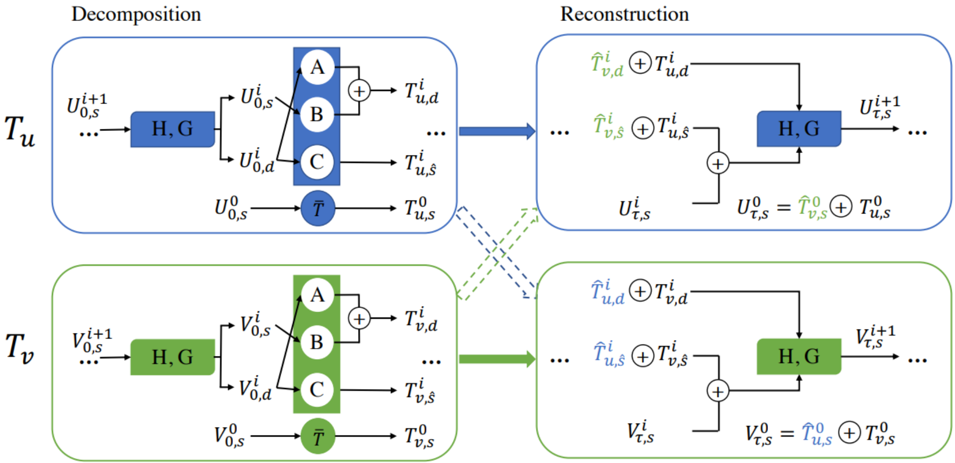


Fig. 10: Enter Caption