Recent Advances in Physics-Informed Machine Learning

AAAI 2024 Tutorial

Yiping Lu (NYU & Northwestern), Grant M. Rotskoff (Stanford)

2024/2

Table of Contents

Scientific Paradigm

Physical Science · · · · · · theoretical derivation combined with experimental verification to study natural phenomena

Numerical Science · · · · · · numerical simulation to understand complex real systems

Physical Science · · · · · · theoretical derivation combined with experimental verification to study natural phenomena

Numerical Science · · · · · · numerical simulation to understand complex real systems

Paul M. Dirac (1929)

"The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are **thus completely known**, and the difficulty is only that the exact application of these laws **leads to equations much too complicated to be soluble**."

Scientific Paradigm

Physical Science · · · · · · theoretical derivation combined with experimental verification to study natural phenomena

Numerical Science · · · · · · numerical simulation to understand complex real systems

Paul M. Dirac (1929)

"The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are **thus completely known**, and the difficulty is only that the exact application of these laws **leads to equations much too complicated to be soluble**."

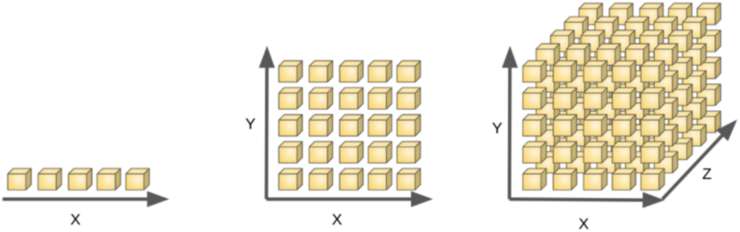
- Accurate "constitutive equation"
  - *e.g.* Newton's gravitational law $\rightarrow$ Kepler's Law
- Efficient algorithms required

Traditional numerical methods approximate general functions using polynomials or piecewise polynomials, however…

Traditional numerical methods approximate general functions using polynomials or piecewise polynomials, however...

Curse of Dimensionality

The cost to represent a function is exponential in the dimensionality.



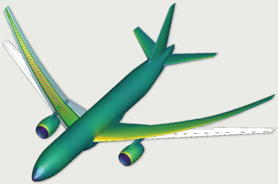
Physical systems *require* high-dimensional representations, *e.g.* dimension of quantum many-body problem $\propto$ # electrons

Inverse Problem/Optimal Design

Inverse problems/optimal design involve solving

$$\min_{x} \mathcal{L}(F(x)),$$

where F is the forward process, such as a physics simulation, and $\mathcal{L}$ is the objective aim to optimize. Even when a single iteration of this forward process is manageable, the overall task becomes computationally infeasible due to the iterative optimization process.



Physical Science · · · · · · theoretical derivation combined with experimental verification to study natural phenomena

Numerical Science · · · · · · numerical simulation to understand complex real systems

Combating these challenges using Machine (Deep) Learning!

Physical Science	theoretical derivation combined with experimental verification to study natural phenomena
Numerical Science	numerical simulation to understand complex real systems
Machine Learning	understand and build models that leverage empirical data to improve performance

Neural Networks provide tools to build flexible, universal, and efficient approximations for complex high-dimensional functions and functionals.

- **In practice**
  - Imagenet (32x32 dimension)
  - Alpha Go (19x19 dimension)
  - Large Language Models ($d_{\text{model}} \sim \mathcal{O}(10^3)$)
- **In theory**
  - Separation to Kernel (Linear) Methods
  - Depth Separation

What is Phyiscs-Informed Machine Learning?

- **Physical Science** · · · · · · theoretical derivation combined with experimental verification to study natural phenomena
- **Numerical Science** · · · · · · numerical simulation to understand complex real systems
- **Machine Learning** · · · · · · understand and build models that leverage empirical data to improve performance
- **Physics-Informed Machine Learning** · · · · · · **TODAY**

Physics-informed Machine Learning study potential benefits for machine learning models by incorporating the physical prior such as

- **Differential Equations**: ODEs, PDEs, S(P)DEs
- **Law of conservation**, **Symmetry** ...

What is Phyiscs-Informed Machine Learning?

- **Physical Science** · · · · · · theoretical derivation combined with experimental verification to study natural phenomena
- **Numerical Science** · · · · · · numerical simulation to understand complex real systems
- **Machine Learning** · · · · · · understand and build models that leverage empirical data to improve performance
- **Physics-Informed Machine Learning** · · · · · · **TODAY**

Physics-informed Machine Learning study potential benefits for machine learning models by incorporating the physical prior such as

- **Differential Equations**: ODEs, PDEs, S(P)DEs
- **Law of conservation**, **Symmetry** ...

Applications include

- Quantum Many-body Problem
- Turbulence Models
- Modeling Rare Events

Hao Z, Liu S, Zhang Y, et al. Physics-informed machine learning: A survey on problems, methods and applications. arXiv preprint arXiv:2211.08064, 2022.

Representation: Higher Dimension

Imagenet only 32 x 32 dimension, which can only simulate $\sim$300 molecules

Thus we need to understand

- function space that we can approximate in high-dimension.
- physical prior can help to represent functions more efficiently.
- approximation theory in infinite dimensional.

What are the Challenges in PIML?

Representation: Higher Dimension

Imagenet only 32 x 32 dimension, which can only simulate $\sim$300 molecules
Thus we need to understand

- function space that we can approximate in high-dimension.
- physical prior can help to represent functions more efficiently.
- approximation theory in infinite dimensional.

Generalization: Expensive Data Collection

Labeling data for scientific research is expansive, thus we need to consider the generalization theory for physics-informed machine learning

Small Data Lots of Physics	$\longleftrightarrow$	Big Data **No Physics**

What is this tutorial about?

How to represent a physical solution and why it generalizes for

- Solving Differential Equations and Optimal Control
- Better Sampling for scientific problems

What is this tutorial about?

How to represent a physical solution and why it generalizes for

- Solving Differential Equations and Optimal Control
- Better Sampling for scientific problems

with applications in

- Inverse Problem
- Quantum Many-body Problem
- Rare Event (Transition Path) Sampling
- Large Deviations

Table of Contents

Physics-Informed Machine Learning

physics-informed machine learning as a structured risk minimization problem

$$\min_{f\in\mathcal{H}} \mathcal{L}(f,\mathcal{D}) + \underbrace{\Omega(f)}_{\text{physical prior}} \tag{1}$$

- **Data** $\mathcal{D}$: we could augment the dataset utilizing available physical prior like symmetry
- **Model** f: we could embed physical prior into the model design
- **Regularization** Ω: regularization terms using given physical priors like differential equations

Physics-Informed Machine Learning

physics-informed machine learning as a structured risk minimization problem

$$\min_{f \in \mathcal{H}} \mathcal{L}(f, \mathcal{D}) + \underbrace{\Omega(f)}_{\text{physical prior}} \tag{1}$$

- **Data** $\mathcal{D}$: we could augment the dataset utilizing available physical prior like symmetry
- **Model** f: we could embed physical prior into the model design
- **Regularization** Ω: regularization terms using given physical priors like differential equations

Tasks that we are interested in

- Solving physical equations (First principle modeling)
- Operator Learning
- System Identification/Scientific Discovery

Hao Z, Liu S, Zhang Y, et al. Physics-informed machine learning: A survey on problems, methods and applications. arXiv preprint arXiv:2211.08064, 2022.

Definition:PDEs

PDE is a relation of the following type, parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, \ldots; \lambda) = 0$$

with suitable boundary conditions

$$\mathcal{B}(u, \vec{x}) = 0, \quad \vec{x} \in \partial\Omega, \tag{2}$$

Solving a PDE $\rightarrow$ find a u function satisfying the governing equation.

where

- $u = u(x_1, \ldots x_n)$ is a unknown function of n variables, i.e., $u : \mathbb{R}^d \mapsto \mathbb{R}$;
  ($\vec{u}$ can be a vector, i.e.,$\vec{u} \in \mathbb{R}^d$, here, we assume it to be a scalar for simplicity)
- $u_{x_i} = \frac{\partial u}{\partial x_i}, u_{x_i x_j} = \frac{\partial^2 u}{\partial x_i x_j}, \ldots$
- f is the governing equation.
- $\mathcal{B}$ is the boundary condition.

Governing equation: Linear vs. Non-linear

Linear PDE

A PDE is linear if and only if f is linear with respect to u and all its derivatives.

$$f(\vec{x}; u; u_{x_1}, \dots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2} \dots; \lambda) = 0, \quad \vec{x} \in \Omega, \tag{3}$$

Governing equation: Linear vs. Non-linear

Linear PDE

A PDE is linear if and only if f is linear with respect to u and all its derivatives.

$$f(\vec{x}; u; u_{x_1}, ..., u_{x_n}; u_{x_1x_1}, u_{x_1x_2} ...; \lambda) = 0, \quad \vec{x} \in \Omega, \tag{3}$$

Non-linear PDE

- **Semilinear PDE** where f is nonlinear only with respect to u but is linear with respect to all its derivatives;
- **Quasi-linear PDE** where f is linear with respect to the highest order derivatives of u;
- **Fully nonlinear PDE** where f is nonlinear with respect to the highest order derivatives of u.

Order of a PDE

The **highest order** of differentiation occurring in the equation is the order of the equation.

$$f(\vec{x}; u; u_{x_1}, \dots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2} \dots; \lambda) = 0, \quad \vec{x} \in \Omega, \tag{4}$$

Governing equation: order of PDEs

Order of a PDE

The **highest order** of differentiation occurring in the equation is the order of the equation.

$$f(\vec{x}; u; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2} \ldots; \lambda) = 0, \quad \vec{x} \in \Omega, \tag{4}$$

Second order PDEs

Most commonly used in engineering applications.

$$au_{xx} + 2bu_{xy} + cu_{yy} + du_x + eu_y + hu = f$$

where $a, \ldots, f$ are smooth (e.g. C^2) functions of x, y.

- **Elliptic**: $b^2 - ac < 0$, e.g., Laplace equation $u_{xx} + u_{yy} = 0$
- **Parabolic**: $b^2 - ac = 0$, e.g., Diffusion equation $u_t - Du_{xx} = 0$
- **Hyperbolic**: $b^2 - ac > 0$, e.g., Wave equation $u_{tt} - c^2 u_{xx} = 0$

- Dirichlet: specifies the boundary value of u: $u|_{\partial\Omega} = f$
- Neumann: specifies the value of the normal derivative of the u: $u_x|_{\partial\Omega} = f$
- Robin: $c_0 u|_{\partial\Omega} + c_1 u_x|_{\partial\Omega} = f$
- Cauchy: Dirichlet and Neumann, i.e, $u|_{\partial\Omega} = f$ and $u_x|_{\partial\Omega} = f$
- Mixed: different location (x) have different boundary condition.

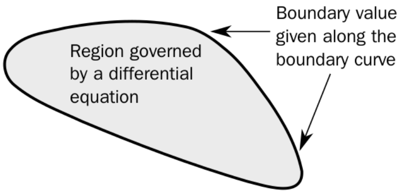


Figure: Boundary value problem [1]

[1] https://en.wikipedia.org/wiki/Boundary_value_problem

Forward problem: Given a fixed λ, solve for $u(x)$

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, \ldots; \lambda) = 0$$

Traditionally, solved with finite difference method (FDM), finite element method (FEM), etc.

Forward problem: Given a fixed λ, solve for $u(x)$

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, \ldots; \lambda) = 0$$

Traditionally, solved with finite difference method (FDM), finite element method (FEM), etc.

Advantages: Accurate, Reliable

Challenges: Expensive and time consuming, Hard to incorporate in downstream applications

Forward problem: Given a fixed λ, solve for $u(x)$

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, \ldots; \lambda) = 0$$

Traditionally, solved with finite difference method (FDM), finite element method (FEM), etc.

Advantages: Accurate, Reliable

Challenges: Expensive and time consuming, Hard to incorporate in downstream applications

Learning is suitable for:

- Surrogate modeling: find a cheap model to surrogate the PDE governed system, i.e., $u = f_\theta(x; \lambda)$.
- Incorporating physical knowledge (PDE) information for downstream tasks.

Inverse problems

Inverse problem: Given a set of observed $u(x)$, find λ

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, \ldots; \lambda) = 0$$

- Identifying unknown parameters in PDEs/boundary/initial conditions
- Data driven (with partail physics knowledge) spatio-temporal modeling

Traditionally, formulated as a PDE constraint optimization problem, and solved with adjoint method.

Inverse problem: Given a set of observed $u(x)$, find λ

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, ..., x_n, ...; u_{x_1}, ..., u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, ...; \lambda) = 0$$

- Identifying unknown parameters in PDEs/boundary/initial conditions
- Data driven (with partail physics knowledge) spatio-temporal modeling

Traditionally, formulated as a PDE constraint optimization problem, and solved with adjoint method.

Learning is suitable for:

- Incorporating PDE information in inverse problems
- Surrogate modeling 1: find a cheap model to surrogate (inversion of) the PDE governed system, i.e., $u = f_\theta(x; \lambda)$ (or $\lambda = f_{\theta'}(u)$).

Most of physics can be formulated as $A(u) = f$ where A is a differential operator

Most of physics can be formulated as $A(u) = f$ where A is a differential operator

PINN solving $A(u) = f$

Suppose we observe $(x_i, f(x_i))_{i=1}^{n}$, can we solve $A(u) = f$?

Most of physics can be formulated as $A(u) = f$ where A is a differential operator

PINN solving $A(u) = f$

Suppose we observe $(x_i, f(x_i))_{i=1}^n$, can we solve $A(u) = f$?
PINN minimizes the equation residual on observed data points, *i.e.*

$$u^* = \arg\min_{u \in \mathcal{H}} \sum_{i=1}^{n} \|A(u)(x_i) - f(x_i)\|^2$$

The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem $\min \|A(u) - f\|$. New transformation can bring new methods!

The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem $\min \|A(u) - f\|$. New transformation can bring new methods!

Deep Ritz Methods Using Variational form, *i.e.* $Ax = b \Longleftrightarrow \min x^\top Ax - 2bx$

Not all PDEs admit a variational form.

Yu B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 2018

The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem $\min \|A(u) - f\|$. New transformation can bring new methods!

Deep Ritz Methods Using Variational form, *i.e.* $Ax = b \iff \min x^\top Ax - 2bx$

Not all PDEs admit a variational form.

Yu B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 2018

Weak Adversarial Network Solving equation $A(u) = f$ equalvalent to

$$\min_u \max_{\|v\| \leq 1} \langle v, A(u) - f \rangle$$

and change the constraint to a log penalization.

Zang Y, Bao G, Ye X, et al. Weak adversarial networks for high-dimensional partial differential equations. Journal of Computational Physics, 2020

Methods Beyond PINN

The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem $\min \|A(u) - f\|$. New transformation can bring new methods!

Deep Ritz Methods Using Variational form, *i.e.* $Ax = b \Longleftrightarrow \min x^\top Ax - 2bx$

Not all PDEs admit a variational form.

Yu B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 2018

Weak Adversarial Network Solving equation $A(u) = f$ equalvalent to

$$\min_u \max_{\|v\| \leq 1} \langle v, A(u) - f \rangle$$

and change the constraint to a log penalization.

Zang Y, Bao G, Ye X, et al. Weak adversarial networks for high-dimensional partial differential equations. Journal of Computational Physics, 2020

Adversarial training L^2 loss is not strong enough (for regualrity of PDE structure), we should use L^∞ loss for some PDEs.

Wang C, et al. Is L^2 Physics Informed Loss Always Suitable for Training Physics Informed Neural Network? Advances in Neural Information Processing Systems, 2022.

A neural network is not the only ansatz (a high bias list...)

- Gaussian Process

Raissi M, et al. Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. SIAM Journal on Scientific Computing, 2018.

Yang S, Wong S W K, Kou S C. Inference of dynamic systems from noisy and sparse data via manifold-constrained Gaussian processes. Proceedings of the National Academy of Sciences, 2021.

Chen Y, Hosseini B, Owhadi H, et al. Solving and learning nonlinear PDEs with Gaussian processes. Journal of Computational Physics, 2021, 447: 110668.

A neural network is not the only ansatz (a high bias list...)

- Gaussian Process

Raissi M, et al. Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. SIAM Journal on Scientific Computing, 2018.

Yang S, Wong S W K, Kou S C. Inference of dynamic systems from noisy and sparse data via manifold-constrained Gaussian processes. Proceedings of the National Academy of Sciences, 2021.

Chen Y, Hosseini B, Owhadi H, et al. Solving and learning nonlinear PDEs with Gaussian processes. Journal of Computational Physics, 2021, 447: 110668.

- Diffusion map

Lai R, Lu J. Point Cloud Discretization of Fokker–Planck Operators for Committor Functions. Multiscale Modeling & Simulation, 2018.

Evans L, et al. Computing committors in collective variables via Mahalanobis diffusion maps. Applied and Computational Harmonic Analysis.

A neural network is not the only ansatz (a high bias list...)

- **Gaussian Process**

  Raissi M, et al. Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. SIAM Journal on Scientific Computing, 2018.

  Yang S, Wong S W K, Kou S C. Inference of dynamic systems from noisy and sparse data via manifold-constrained Gaussian processes. Proceedings of the National Academy of Sciences, 2021.

  Chen Y, Hosseini B, Owhadi H, et al. Solving and learning nonlinear PDEs with Gaussian processes. Journal of Computational Physics, 2021, 447: 110668.

- **Diffusion map**

  Lai R, Lu J. Point Cloud Discretization of Fokker–Planck Operators for Committor Functions. Multiscale Modeling & Simulation, 2018.

  Evans L, et al. Computing committors in collective variables via Mahalanobis diffusion maps. Applied and Computational Harmonic Analysis.

- **Tensor Network**

  Bachmayr M, Schneider R, Uschmajew A. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. Foundations of Computational Mathematics 2016.

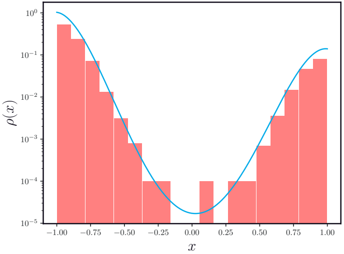
  Richter L, Sallandt L, Nüsken N. Solving high-dimensional parabolic PDEs using the tensor train format International Conference on Machine Learning, 2021.

  Hur Y, Hoskins J G, Lindsey M, et al. Generative modeling via tensor train sketching. Applied and Computational Harmonic Analysis, 2023.

So far, we have only built loss functions. How should we sample data?

$$\mathcal{I}(u) = \int_{\Omega} \mathcal{L}(\boldsymbol{x}, u) d\nu(\boldsymbol{x}),$$

- Simplest case $d\nu(\boldsymbol{x}) = d\boldsymbol{x}$
- Challenging case $d\nu(\boldsymbol{x}) = e^{-U(\boldsymbol{x})} d\boldsymbol{x}$



On-the-fly variance reduction via adaptive importance sampling

$$\mathcal{L}(\boldsymbol{x}, f) = \frac{1}{2}|\nabla f(\boldsymbol{x})|^2, \qquad d\nu(\boldsymbol{x}) = e^{-\beta V(\boldsymbol{x})} d\boldsymbol{x} \qquad (\beta > 0)$$

- Idea 1: Window sampling with instantaneous solution f with a bias
- Idea 2: Reweight the expectation with these importance sampled points

$W_l(\boldsymbol{x}) \geq 0$ with $l = 1, \ldots, L$ such that $\forall \boldsymbol{x} \in \Omega \ : \ \sum_{l=1}^{L} W_l(\boldsymbol{x}) = 1$,

$$\mathbb{E}_\nu \phi = \sum_{l=1}^{L} \int_{\mathbb{R}^d} \phi(\boldsymbol{x}) W_l(\boldsymbol{x}) d\nu(\boldsymbol{x}) \equiv \sum_{l=1}^{L} w_l \, \mathbb{E}_l \phi$$

GM Rotskoff, AR Mitchell, E Vanden-Eijnden Mathematical and Scientific Machine Learning, 757-780

For any ϕ, define

$$\mathbb{E}_l \phi = Z_l^{-1} \int_{\mathbb{R}^d} \phi(\boldsymbol{x}) W_l(\boldsymbol{x}) d\nu(\boldsymbol{x}) \quad \text{where} \quad Z_l = \int_{\mathbb{R}^d} W_l(\boldsymbol{x}) d\nu(\boldsymbol{x}) \tag{5}$$

as well as the weights

$$w_l = \mathbb{E}_\nu W_l. \tag{6}$$

By choosing $\phi(\boldsymbol{x}) = W_{l'}(\boldsymbol{x})$ in this expression, we deduce that the weights satisfy the eigenvalue problem (Thiede et al 2016)

$$w_{l'} = \sum_{l=1}^{L} w_l p_{ll'}, \quad l' = 1, \dots, L, \quad \text{subject to} \quad \sum_{l=1}^{L} w_l = 1, \tag{7}$$

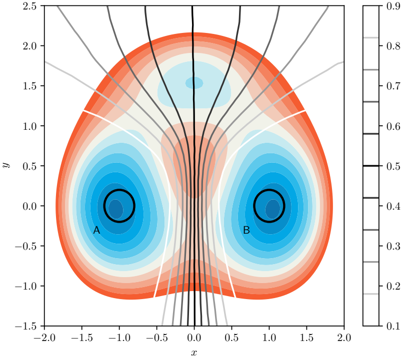
where we defined

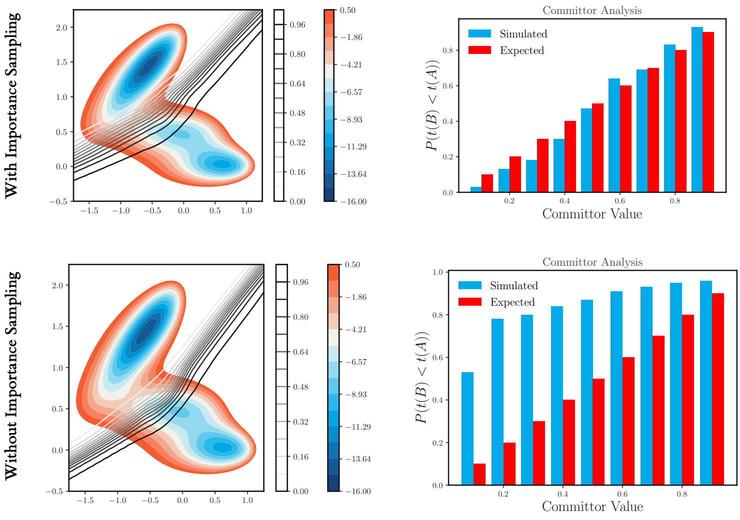
$$p_{ll'} = \langle W_{l'} \rangle_l. \tag{8}$$

Sample $Z_l^{-1} W_l(\boldsymbol{x}) d\nu(\boldsymbol{x})$ with MCMC biased by $-\log W_l(\boldsymbol{x})$ so that

$$\mathbb{E}_l \phi \approx \frac{1}{n} \sum_{i=1}^{n} \phi(\boldsymbol{x}_{i,l}), \qquad \boldsymbol{x}_{i,l} \sim Z_l^{-1} W_l(\boldsymbol{x}) d\nu(\boldsymbol{x})$$

This allows us to estimate $\mathbb{E}_l \phi$ in (7) as well as $p_{ll'}$ in (8) — can solve eigenvalue problem! the weights w_l, and finally estimate $\mathbb{E}_\nu \phi$.




With Importance Sampling
Without Importance Sampling
Committor Analysis
Simulated
Expected
$P(t(B) < t(A))$
Committor Value

Examples of High-Dimensional PDEs

Important Examples of high-dimensional problems

- **Optimal Control**: Hamilton-Jacobi Equation
  - Dimension: State Space Dimension

Examples of High-Dimensional PDEs

Important Examples of high-dimensional problems

- **Optimal Control**: Hamilton-Jacobi Equation
  - Dimension: State Space Dimension
- **Metastability**: Backward Kolmogorov / Feynman-Kac
  - Dimension: State Space Dimension

Important Examples of high-dimensional problems

- **Optimal Control**: Hamilton-Jacobi Equation
  - Dimension: State Space Dimension
- **Metastability**: Backward Kolmogorov / Feynman-Kac
  - Dimension: State Space Dimension
- **Nonequilibrium Dynamics**: Compute large deviation function
  - Dimension: State Space Dimension

Important Examples of high-dimensional problems

- **Optimal Control**: Hamilton-Jacobi Equation
  — Dimension: State Space Dimension
- **Metastability**: Backward Kolmogorov / Feynman-Kac
  — Dimension: State Space Dimension
- **Nonequilibrium Dynamics**: Compute large deviation function
  — Dimension: State Space Dimension
- **Quantum Many-body problem**: Eigenvalue Problems
  — dimension $\propto$ number of particles

Dynamics driven by an SDE,

$$dX_t = -\nabla V(X_t)dt + \sqrt{2\beta^{-1}}dW_t.$$

Various quantities satisfy Backward Kolmogorov Equation, including the *committor* probability:

$$q(\boldsymbol{x}) := \mathbb{P}^{\boldsymbol{x}}(t_B < t_A)$$

where $t_A = \inf\{t : \boldsymbol{x}(t) \in A\}$ and t_B is defined analogously. GM Rotskoff, AR Mitchell, E Vanden-Eijnden. Active importance sampling for variational objectives dominated by rare events: Consequences for optimization and generalization. Mathematical and Scientific Machine Learning, 757-780, 2022.

We want to solve the PDE,

$$\begin{cases} (Lq)(\boldsymbol{x}) = 0 & \text{for } \boldsymbol{x} \notin A \cup B \\ q(\boldsymbol{x}) = 0 & \text{for } \boldsymbol{x} \in A \\ q(\boldsymbol{x}) = 1 & \text{for } \boldsymbol{x} \in B. \end{cases}$$

where $-L$ is the infinitesimal generator of the process defined by (**??**):

$$Lq = \nabla V \cdot \nabla q - \beta^{-1} \Delta q.$$

We want to solve the PDE,

$$\begin{cases} (Lq)(\boldsymbol{x}) = 0 & \text{for } \boldsymbol{x} \notin A \cup B \\ q(\boldsymbol{x}) = 0 & \text{for } \boldsymbol{x} \in A \\ q(\boldsymbol{x}) = 1 & \text{for } \boldsymbol{x} \in B. \end{cases}$$

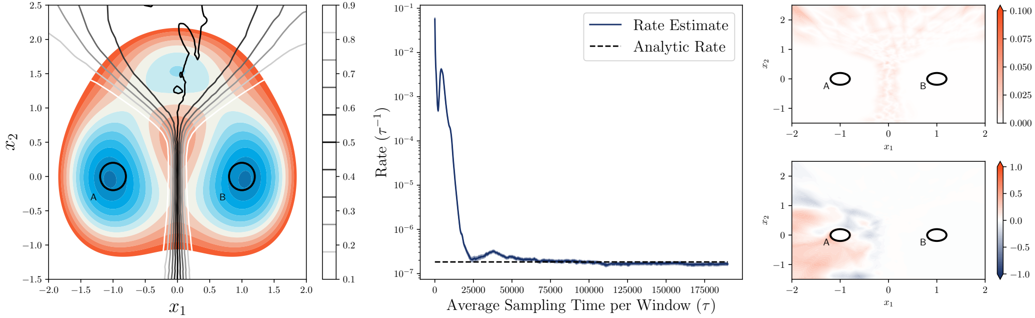
where $-L$ is the infinitesimal generator of the process defined by (**??**):

$$Lq = \nabla V \cdot \nabla q - \beta^{-1} \Delta q.$$

Variational formulation:

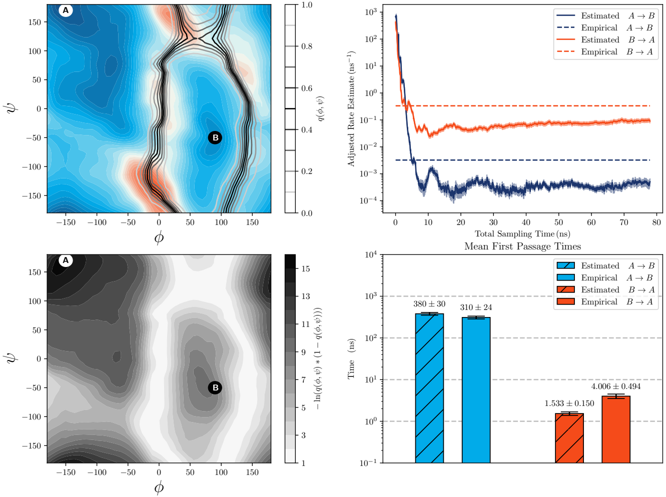
$$C(q) = \int_{\mathbb{R}^d} |\nabla q(\boldsymbol{x})|^2 d\nu(\boldsymbol{x}) \qquad \text{with} \quad d\nu(\boldsymbol{x}) = Z^{-1} e^{-\beta V(\boldsymbol{x})} d\boldsymbol{x}$$

Example: low-dimensional metastable system



Example: 66-dimensional molecular systems

3 Differential Equation Solving



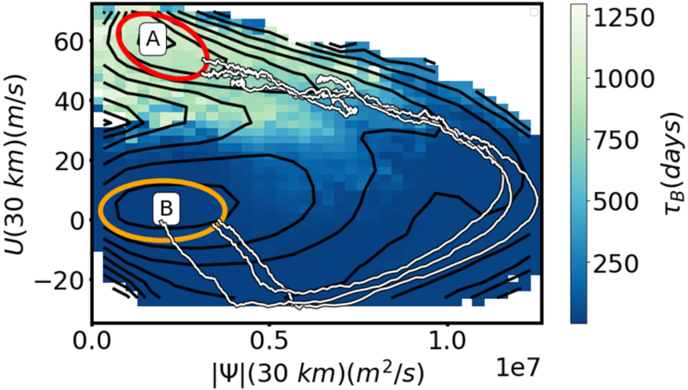
Molecular dynamics becomes Markovian on a lag-time τ.

$$
\begin{aligned}
\mathcal{T}^{\tau}_{A\cup B}[\phi](\boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{x}}\phi(\boldsymbol{X}_{\tau_*}) \quad \text{where} \quad \tau_* = \min(\tau, T) \\
q(\boldsymbol{x}) &= 0 \quad \boldsymbol{x} \in A \\
q(\boldsymbol{x}) &= 1 \quad \boldsymbol{x} \in B
\end{aligned}
$$

Then, the committor satisfies:

$$
(\mathcal{T}^{\tau}_{A\cup B}[q] - \mathsf{Id})(\boldsymbol{x}) = 0 \quad \boldsymbol{x} \in (A \cup B)^{\mathrm{c}}
$$

Strahan, J. et al. http://arxiv.org/abs/2208.01717 (2022).



Strahan, J. et al. http://arxiv.org/abs/2208.01717 (2022).

Aim to calculate the large deviation rate function for an observable, giving information about the asymptotic probability distribution.

Large deviations on path measures

$$\mathbb{P}(A_T \in [a, a + da]) \asymp e^{-TI(a)}$$

where $\mathbb{P}$ is the path measure associated with

$$dX_t = b(X_t)dt + \sigma(X_t)dW_t$$

and

$$A_T = \int_0^T f(X_t)dt + \int_0^T g(X_t) \circ dX_t$$

J Yan, GM Rotskoff. Physics-informed graph neural networks enhance scalability of variational nonequilibrium optimal control. Journal of Chemical Physics 157 (7)

Control problem hidden in a rare events problem

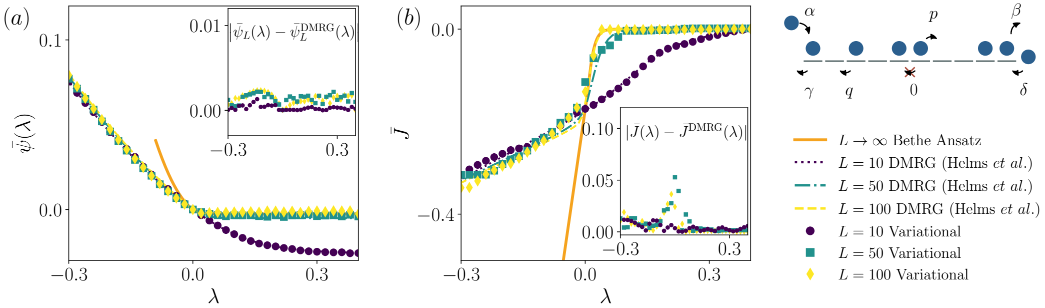
$$d\boldsymbol{X}_t^u = u_t(\boldsymbol{X}_t^u)dt + \sigma d\boldsymbol{W}_t, \tag{9}$$

and then the SCGF can be estimated simply by reweighting the average

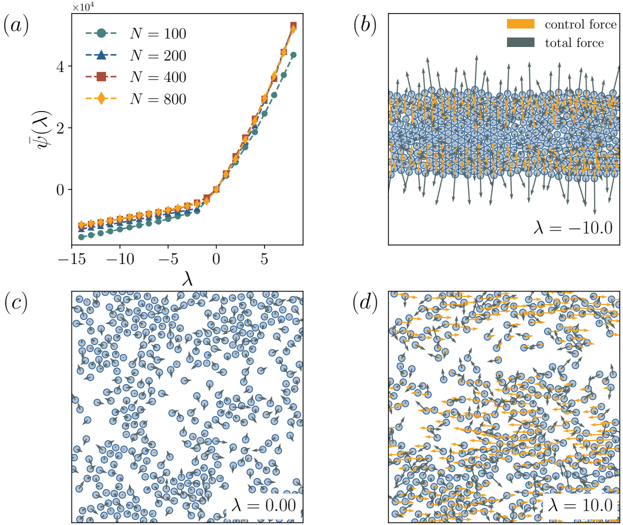
$$\psi(\lambda) = \lim_{T\to\infty} \frac{1}{T} \log \mathbb{E}_{\boldsymbol{X}^u} \left(e^{\lambda T A_T} \frac{\mathrm{d}\mathbb{P}[\boldsymbol{X}^u]}{\mathrm{d}\mathbb{P}_u[\boldsymbol{X}^u]} \right). \tag{10}$$

$$\psi(\lambda) = \lim_{T\to\infty} \frac{1}{T} \log \mathbb{E}[e^{\lambda T A_T}] = \sup_u \lim_{T\to\infty} \left\{ \lambda \mathbb{E}_u[A_T] - \frac{1}{T} \mathcal{D}_{KL}[\mathrm{d}\mathbb{P}_u \| \mathrm{d}\mathbb{P}] \right\}. \tag{11}$$

Current in the asymmetric exclusion process; comparison with tensor network approach.



Active nonequilibrium matter, quantifying entropy production fluctuations.



Quantum Monte Carlo aims to calculate the wave function (eigenfunction). Monte Carlo here means handle the multi-dimensional integrals that arise in the different formulations of the many-body problem.

Quantum Monte Carlo aims to calculate the wave function (eigenfunction). Monte Carlo here means handle the multi-dimensional integrals that arise in the different formulations of the many-body problem.

Variational Monte Carlo

We can recast the problem that finding the eigenvalue of $\mathcal{H}$ as

$$\min_{\theta} \mathcal{L}(\theta) = \frac{\langle \Psi_\theta, \mathcal{H}\Psi_\theta \rangle}{\langle \Psi_\theta, \Psi_\theta \rangle} = \frac{\int_{x \in \mathcal{X}} \Psi_\theta^*(x) \cdot (\mathcal{H}\Psi_\theta)(x) dx}{\int_{x \in \mathcal{X}} \Psi_\theta^*(x) \cdot \Psi_\theta(x) dx}$$

Quantum Monte Carlo aims to calculate the wave function (eigenfunction). Monte Carlo here means handle the multi-dimensional integrals that arise in the different formulations of the many-body problem.

Variational Monte Carlo

We can recast the problem that finding the eigenvalue of $\mathcal{H}$ as

$$\begin{aligned}\min_{\theta} \mathcal{L}(\theta) &= \frac{\langle \Psi_\theta, \mathcal{H}\Psi_\theta \rangle}{\langle \Psi_\theta, \Psi_\theta \rangle} = \frac{\int_{x\in\mathcal{X}} \Psi_\theta^*(x) \cdot (\mathcal{H}\Psi_\theta)(x) dx}{\int_{x\in\mathcal{X}} \Psi_\theta^*(x) \cdot \Psi_\theta(x) dx} \\ &= \int_{x\in\mathcal{X}} \underbrace{\frac{\Psi_\theta^*(x) \cdot \Psi_\theta(x)}{\int_{x\in\mathcal{X}} \Psi_\theta^*(x) \cdot \Psi_\theta(x) dx}}_{\text{probability } \pi_\theta(x)} \underbrace{\frac{\mathcal{H}\Psi_\theta(x)}{\Psi_\theta(x)}}_{\text{local energy } E_\theta(x)} dx = \mathbb{E}_{\pi_\theta(x)} E_\theta(x)\end{aligned} \tag{12}$$

Variational Monte Carlo

- Fisher-Rao Gradient

$$\nabla_\theta \mathcal{L}_\theta = 2\mathbb{E}_{\pi_\theta(x)}\Big[\underbrace{\big(E_\theta(x) - \mathbb{E}_{\pi_\theta(x)} E_\theta(x)\big)}_{\text{deviation of local energy}} \underbrace{\nabla_\theta \log|\Psi_\theta|}_{\text{score}}\Big]$$

Pfau D, Spencer J S, et al. Ab initio solution of the many-electron Schrödinger equation with deep neural networks. Physical Review Research, 2020, 2(3): 033429.

Variational Monte Carlo

- Fisher-Rao Gradient

$$\nabla_\theta \mathcal{L}_\theta = 2\mathbb{E}_{\pi_\theta(x)}\Big[\underbrace{\big(E_\theta(x) - \mathbb{E}_{\pi_\theta(x)} E_\theta(x)\big)}_{\text{deviation of local energy}} \underbrace{\nabla_\theta \log|\Psi_\theta|}_{\text{score}}\Big]$$

Pfau D, Spencer J S, et al. Ab initio solution of the many-electron Schrödinger equation with deep neural networks. Physical Review Research, 2020, 2(3): 033429.

- Wasserstein Gradient

$$\nabla_\theta \mathcal{L}_\theta = \mathbb{E}_\theta \nabla_\theta \left\langle -2 \underbrace{\nabla_x E_{\mathrm{loc}}(x^{(i)})}_{\text{gradient of local energy}}, \nabla_x \log \pi(x^{(i)}, \theta) \right\rangle$$

Neklyudov K, Nys J, Thiede L, et al. Wasserstein quantum monte carlo: A novel approach for solving the quantum many-body schrödinger equation. Neurips 2023.

- Design of ani-symmetric Neural Network $\Psi(x_{\sigma(1)}, \cdots, x_{\sigma(n)}) = \text{sgn}(\sigma)\Psi(x_1, \cdots, x_n)$
  (Slater) Determinant is slow and exists an exponential approximation lower bound.

  Zweig A, Bruna J. Towards Antisymmetric Neural Ansatz Separation. arXiv preprint arXiv:2208.03264, 2022.

  Pang T, Yan S, Lin M. $O(N^2)$ Universal Antisymmetry in Fermionic Neural Networks. arXiv preprint arXiv:2205.13205, 2022.

Why VMC is hard?

- Design of ani-symmetric Neural Network $\Psi(x_{\sigma(1)}, \cdots, x_{\sigma(n)}) = \mathrm{sgn}(\sigma)\Psi(x_1, \cdots, x_n)$ (Slater) Determinant is slow and exists an exponential approximation lower bound.

  Zweig A, Bruna J. Towards Antisymmetric Neural Ansatz Separation. arXiv preprint arXiv:2208.03264, 2022.

  Pang T, Yan S, Lin M. $O(N^2)$ Universal Antisymmetry in Fermionic Neural Networks. arXiv preprint arXiv:2205.13205, 2022.

- The calculation of Δ is slow in high dimension

$$\mathcal{H} := -\frac{1}{2}\sum_{i}\Delta_i + \sum_{i>j}\frac{1}{|r_i - r_j|} - \sum_{iI}\frac{Z_I}{|r_i - R_I|} + \sum_{I>J}\frac{Z_I Z_J}{|R_I - R_J|} \tag{13}$$

Li R, Ye H, Jiang D, et al. Forward Laplacian: A New Computational Framework for Neural Network-based Variational Monte Carlo. arXiv preprint arXiv:2307.08214, 2023.

Why VMC is hard?

- Design of ani-symmetric Neural Network $\Psi(x_{\sigma(1)}, \cdots, x_{\sigma(n)}) = \text{sgn}(\sigma)\Psi(x_1, \cdots, x_n)$ (Slater) Determinant is slow and exists an exponential approximation lower bound.

  Zweig A, Bruna J. Towards Antisymmetric Neural Ansatz Separation. arXiv preprint arXiv:2208.03264, 2022.

  Pang T, Yan S, Lin M. $O(N^2)$ Universal Antisymmetry in Fermionic Neural Networks. arXiv preprint arXiv:2205.13205, 2022.

- The calculation of Δ is slow in high dimension

$$\mathcal{H} := -\frac{1}{2}\sum_{i}\Delta_i + \sum_{i>j}\frac{1}{|r_i - r_j|} - \sum_{iI}\frac{Z_I}{|r_i - R_I|} + \sum_{I>J}\frac{Z_I Z_J}{|R_I - R_J|} \tag{13}$$

  Li R, Ye H, Jiang D, et al. Forward Laplacian: A New Computational Framework for Neural Network-based Variational Monte Carlo. arXiv preprint arXiv:2307.08214, 2023.

- Non-convex landscape and overfitting

  Zhang H, Webber R J, Lindsey M, et al. Understanding and eliminating spurious modes in variational Monte Carlo using collective variables. Physical Review Research, 2023, 5(2): 023101.

PINN-like idea can be used beyond physics. Generally speaking, you can always fusion modeling with learning with PINN, for example

- Auction Design

  Dütting P, Feng Z, Narasimhan H, et al. Optimal auctions through deep learning International Conference on Machine Learning.

  Peri N, Curry M, Dooley S, et al. Preferencenet: Encoding human preferences in auction design with deep learning. Advances in Neural Information Processing Systems 2021.

PINN-like idea can be used beyond physics. Generally speaking, you can always fusion modeling with learning with PINN, for example

- Auction Design

  Dütting P, Feng Z, Narasimhan H, et al. Optimal auctions through deep learning International Conference on Machine Learning.

  Peri N, Curry M, Dooley S, et al. Preferencenet: Encoding human preferences in auction design with deep learning. Advances in Neural Information Processing Systems 2021.

- Neural Rendering

  Mildenhall B, Srinivasan P P, Tancik M, et al. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM.

  Sitzmann V, Martel J, et al. Implicit neural representations with periodic activation functions. Advances in neural information processing systems 2020.

- . . .

Table of Contents

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?
Is PINN Optimal?

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

Recast Solving PDE as a Statistical Problem

Example: Solving $\Delta u = f$

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

Recast Solving PDE as a Statistical Problem

Example: Solving $\Delta u = f$

- Hypothesis Space: the solution u in (Sobolev, Besov, Barron space...)

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

Recast Solving PDE as a Statistical Problem

Example: Solving $\Delta u = f$

- Hypothesis Space: the solution u in (Sobolev, Besov, Barron space...)
- Observation Data:

$$(u(x_i), f(x_i) = \Delta u(x_i) + \text{noise})_{i=1}^{n}$$

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

Recast Solving PDE as a Statistical Problem

Example: Solving $\Delta u = f$

- Hypothesis Space: the solution u in (Sobolev, Besov, Barron space...)
- Observation Data:

$$(u(x_i), f(x_i) = \Delta u(x_i) + \text{noise})_{i=1}^{n}$$

Now we recast a solving PDE problem as a non-parametric estimation problem, so that we can

- Using Fano, ... methods to know the lower bound
- Using empirical process, ... methods to build the upper bound

Lu, Yiping, et al. "Machine learning for elliptic pdes: Fast rate generalization bound, neural scaling law and minimax optimality."

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

Information Theortical Lower Bound

If A is a $t-$th order linear differential operator, then any Estimator H using $(X_i, f_i)_{i=1}^n$ can't do better than

$$\inf_{H} \sup_{u \in C^{\alpha}(\Omega)} \mathbb{E} \| H(\{X_i, f_i\}_{i=1,\cdots,n}) - u^* \|_{W_s^2} \gtrsim n^{-\frac{2\alpha - 2s}{2\alpha - 2t + d}},$$

- Solving a PDE equal to reconstructing a function with gradient information

inf means best estimator and sup means the hardest problem

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?
Is PINN Optimal? Are All Losses Created Equal?

Information Theortical Lower Bound

If A is a $t-$th order linear differential operator, then any Estimator H using $(X_i, f_i)_{i=1}^n$ can't do better than

$$\inf_{H} \sup_{u \in C^\alpha(\Omega)} \mathbb{E}\|H(\{X_i, f_i\}_{i=1,\cdots,n}) - u^*\|_{W_s^2} \gtrsim n^{-\frac{2\alpha - 2s}{2\alpha - 2t + d}},$$

- Solving a PDE equal to reconstructing a function with gradient information

inf means best estimator and sup means the hardest problem

Take Home Message PINN is Optimal!

What is the optimal sample complexity for learning with prior information $A(u) = f$?
Is PINN Optimal? Are All Losses Created Equal?

Information Theortical Lower Bound

If A is a $t-$th order linear differential operator, then any Estimator H using $(X_i, f_i)_{i=1}^n$ can't do better than

$$\inf_{H} \sup_{u \in C^{\alpha}(\Omega)} \mathbb{E}\|H(\{X_i, f_i\}_{i=1,\cdots,n}) - u^*\|_{W_s^2} \gtrsim n^{-\frac{2\alpha-2s}{2\alpha-2t+d}},$$

- Solving a PDE equal to reconstructing a function with gradient information

inf means best estimator and sup means the hardest problem

Take Home Message PINN is Optimal! Not every consistent loss function is optimal! We need case by case studying!

Lu, Yiping, et al. "Machine learning for elliptic pdes: Fast rate generalization bound, neural scaling law and minimax optimality."

Why Deep Ritz Method is sub-optimal?

A Fourier Basis View

Why Deep Ritz Method is sub-optimal? Solving a simple PDE $\Delta u = f$ using Fourier Basis. Using Deep Ritz Methods, the objective function is

$$\min \int \frac{1}{2}\|\nabla f(x)\|^2 - u(x)f(x)dx$$

Why Deep Ritz Method is sub-optimal? Solving a simple PDE $\Delta u = f$ using Fourier Basis. Using Deep Ritz Methods, the objective function is

$$\min \int \frac{1}{2}||\nabla f(x)||^2 - u(x)f(x)dx$$

- **Estimator 1:** First learn f, the god solves the equation computationally intractable

Estimator 1

First Estimate f then solve u, $f_z = \frac{1}{n}\sum f(x_i)\phi_z(x_i)$, then $u = \sum \frac{1}{||z||^2} f_z\phi_z(x)$

Why Deep Ritz Method is sub-optimal? Solving a simple PDE $\Delta u = f$ using Fourier Basis. Using Deep Ritz Methods, the objective function is

$$\min \int \frac{1}{2}\|\nabla f(x)\|^2 - u(x)f(x)dx$$

- **Estimator 1**: First learn f, the god solves the equation computationally intractable

Estimator 1

First Estimate f then solve u, $f_z = \frac{1}{n}\sum f(x_i)\phi_z(x_i)$, then $u = \sum \frac{1}{\|z\|^2} f_z \phi_z(x)$

- **Estimator 2**: Deep ritz methods

Estimator 2

Plug $u = \sum u_z \phi_z(x)$ into the Deep Ritz Objective function

$$\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{z} u_z \nabla\phi_z(x_i)\right)^2 + \sum_{z} u_z\phi_z(x_i)f(x_i)$$

- **Estimator 1:** The Fourier coefficient of the solution of Estimator 1 is

$$\mathrm{u}_{1,z} = \underbrace{\operatorname{diag}\left(\|z\|_2^2\right)^{-1}_{\|z\|_\infty \le Z}}_{\text{Matrix}A} f_z. \tag{14}$$

- **Estimator 2:** The Fourier coefficient of the solution of Estimator 2 is

$$\mathrm{u}_{2,z} = \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \nabla\phi_i(x_i)\nabla\phi_j(x_i)\right)^{-1}_{\|i\|_\infty \le Z, \|j\|_\infty \le Z}}_{\text{Empirical Gram Matrix } \hat{A}} f_z, \tag{15}$$

- **Estimator 1:** The Fourier coefficient of the solution of Estimator 1 is

$$u_{1,z} = \underbrace{\text{diag}\left(\|z\|_2^2\right)^{-1}_{\|z\|_\infty \le Z}}_{\text{Matrix}A} f_z. \tag{14}$$

- **Estimator 2:** The Fourier coefficient of the solution of Estimator 2 is

$$u_{2,z} = \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} \nabla\phi_i(x_i)\nabla\phi_j(x_i)\right)^{-1}_{\|i\|_\infty \le Z, \|j\|_\infty \le Z}}_{\text{Empirical Gram Matrix } \hat{A}} f_z, \tag{15}$$

Suboptimality of Deep Ritz Methods

introduce a new variance $\text{Var}(\|\nabla u(x)\|^2 - \Delta u(x)u(x))$, **but neglectable in high-dimension**

Lu, Yiping, et al. "Machine learning for elliptic pdes: Fast rate generalization bound, neural scaling law and minimax optimality."

Question

Let's consider the simplest PDE $\Delta u = f$. If f can be represented by a NN, can u be represented by a NN?

Question

Let's consider the simplest PDE $\Delta u = f$. If f can be represented by a NN, can u be represented by a NN?

The answer is **YES**. This helps us to understand the implicit bias of NN to solve PDEs.

- Parametric Complexity Bounds for Approximating PDEs with Neural Networks Tanya Marwah, Zachary C. Lipton, Andrej Risteski Neural Information Processing Systems (NeurIPS), 2021
- Neural Network approximations of PDEs Beyond Linearity: A Representational Perspective Tanya Marwah, Zachary C. Lipton, Jianfeng Lu, Andrej Risteski International Conference on Machine Learning (ICML), 2023
- Deep Equilibrium Based Neural Operators for Steady-State PDEs Tanya Marwah*, Ashwini Pokle*, J. Zico Kolter, Zachary C. Lipton, Jianfeng Lu, Andrej Risteski Neural Information Processing Systems (NeurIPS), 2023

Question

Let's consider the simplest PDE $\Delta u = f$. If f can be represented by a NN, can u be represented by a NN?

The answer is **YES**. This helps us to understand the implicit bias of NN to solve PDEs.

- Parametric Complexity Bounds for Approximating PDEs with Neural Networks Tanya Marwah, Zachary C. Lipton, Andrej Risteski Neural Information Processing Systems (NeurIPS), 2021
- Neural Network approximations of PDEs Beyond Linearity: A Representational Perspective Tanya Marwah, Zachary C. Lipton, Jianfeng Lu, Andrej Risteski International Conference on Machine Learning (ICML), 2023
- Deep Equilibrium Based Neural Operators for Steady-State PDEs Tanya Marwah*, Ashwini Pokle*, J. Zico Kolter, Zachary C. Lipton, Jianfeng Lu, Andrej Risteski Neural Information Processing Systems (NeurIPS), 2023

Reason: Neural Network can perform (preconditioned) gradient flow.

- Similar to the recent line of that transformer can perform gradient descent for in-context learning.
- Precondition is essential for infinite-dimensional due to infinite condition number!

Will different loss function affects optimization speed?

1).**Physics-Informed** $\int(\Delta u(x) - f(x))^2 dx$ 2).**Deep Ritz** $\int \|\Delta u(x)\|^2 - 2u(x)f(x)dx$

Will different loss function affects optimization speed?

1).**Physics-Informed** $\int (\Delta u(x) - f(x))^2 dx$ 2).**Deep Ritz** $\int \|\Delta u(x)\|^2 - 2u(x)f(x)dx$

Traditional Thoughts 1) is much harder, for it involves condition number of Δ^2 while 2) only involves Δ

Will different loss function affects optimization speed?

1).**Physics-Informed** $\int (\Delta u(x) - f(x))^2 dx$ 2).**Deep Ritz** $\int \|\Delta u(x)\|^2 - 2u(x)f(x)dx$

Traditional Thoughts 1) is much harder, for it involves condition number of Δ^2 while 2) only involves Δ

Machine Learning is a Kernelized gradient flow. Physics equations can precondition machine learning!

Lu Y, Blanchet J, Ying L. Sobolev acceleration and statistical optimality for learning elliptic equations via gradient descent. Advances in Neural Information Processing Systems, 2022, 35: 33233-33247.

Will different loss function affects optimization speed?

1).**Physics-Informed** $\int (\Delta u(x) - f(x))^2 dx$ 2).**Deep Ritz** $\int \|\Delta u(x)\|^2 - 2u(x)f(x)dx$

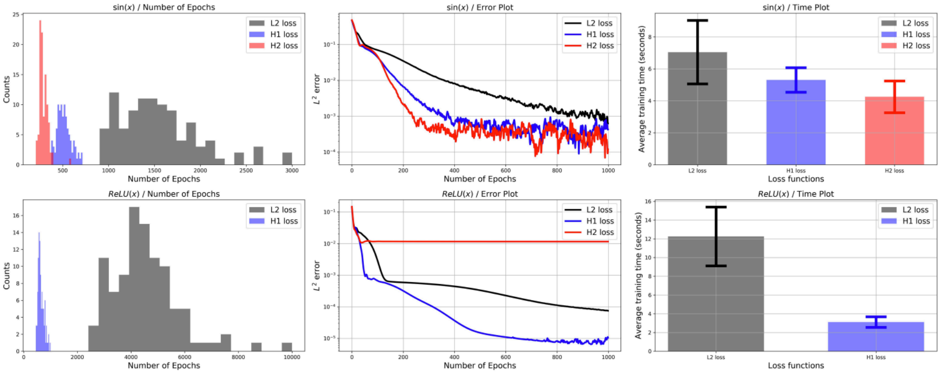
Traditional Thoughts 1) is much harder, for it involves condition number of Δ^2 while 2) only involves Δ

Machine Learning is a Kernelized gradient flow. Physics equations can precondition machine learning!

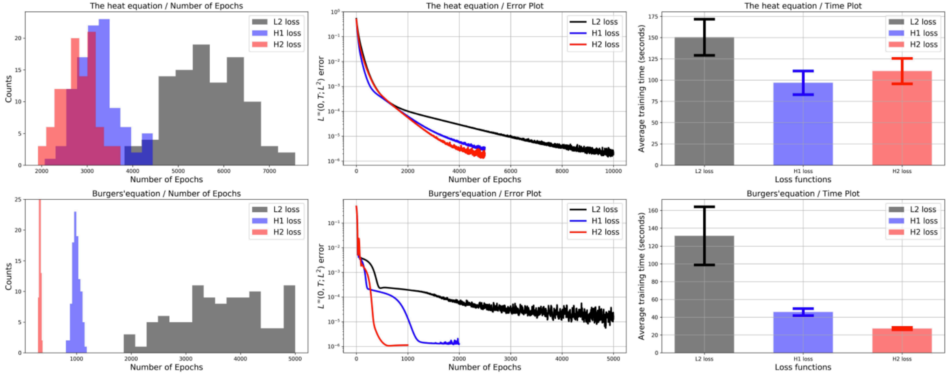
Lu Y, Blanchet J, Ying L. Sobolev acceleration and statistical optimality for learning elliptic equations via gradient descent. Advances in Neural Information Processing Systems, 2022, 35: 33233-33247.

Using Sobolev norm ($\int \|\nabla^k(\Delta u(x) - f(x))\|^2 dx$) as loss function can further accelerates training accelerates optimization

- Yu J, Lu L, Meng X, et al. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, 2022.
- Sobolev training for physics-informed neural networks, with J. W. Jang, W. J. Han, and H. J. Hwang, 2023



Sobolev Training vs L2 training for function fitting.



Sobolev Training vs L2 training for solving heat equation.

Computation of PINN in High Dimension

Computing and even back prop $\Delta u = \underbrace{u_{x_1 x_1} + \cdots + u_{x_d x_d}}_{d \text{ times computation}}$ is hard when d is high!

Computing and even back prop $\Delta u = \underbrace{u_{x_1 x_1} + \cdots + u_{x_d x_d}}_{d \text{ times computation}}$ is hard when d is high!

Idea 1: Stein's Lemma: $u = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} f(x+\delta)$, then $\nabla_x u = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} [\frac{\delta}{\sigma^2} f(x+\delta)]$

- Relates to Feyman-Kac
- Finite Difference with random direction!

He D, Li S, Shi W, et al. Learning physics-informed neural networks without stacked back-propagation International Conference on Artificial Intelligence and Statistics.

Computing and even back prop $\Delta u = \underbrace{u_{x_1 x_1} + \cdots + u_{x_d x_d}}_{d \text{ times computation}}$ is hard when d is high!

Idea 1: Stein's Lemma: $u = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)} f(x+\delta)$, then $\nabla_x u = \mathbb{E}_{\delta \sim \mathcal{N}(0,\sigma^2 I)}[\frac{\delta}{\sigma^2} f(x+\delta)]$

- Relates to Feyman-Kac
- Finite Difference with random direction!

He D, Li S, Shi W, et al. Learning physics-informed neural networks without stacked back-propagation International Conference on Artificial Intelligence and Statistics.

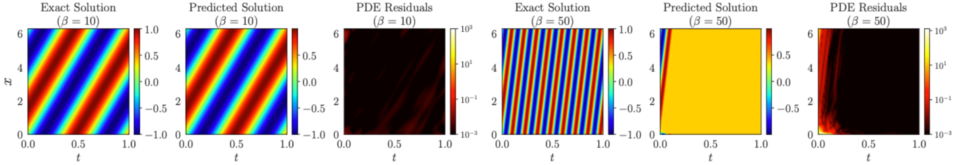
Idea 2: Sketching: random select dimension to descent

$$\Delta u(x) = \mathbb{E}_i \frac{d^2}{dx_i^2} u(x)$$

Hu Z, Shukla K, Karniadakis G E, et al. Tackling the curse of dimensionality with physics-informed neural networks. arXiv preprint arXiv:2307.12306, 2023.

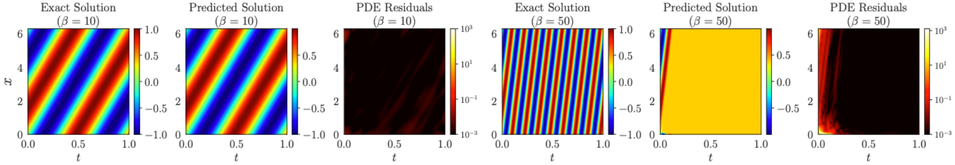
Failure Modes of PINN

Consider solving equation $\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0$ whose solution is $u(x,t) = u(x - \beta t, 0)$ using PINN



Propagation Failure: some collocation points start converging to trivial solutions before the correct solution from initial/boundary points is able to reach them

Consider solving equation $\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0$ whose solution is $u(x,t) = u(x - \beta t, 0)$ using PINN



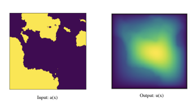
Propagation Failure: some collocation points start converging to trivial solutions before the correct solution from initial/boundary points is able to reach them

- Curriculum training using easier β Krishnapriyan A, et al. Characterizing possible failure modes in physics-informed neural networks. Neurips, 2021.
- Respecting causality Wang S, et al. Respecting causality is all you need for training physics-informed neural networks. arXiv:2203.07404.
- Adaptive sampling Gao Z, Yan L, Zhou T. Failure-informed adaptive sampling for PINNs. SIAM Journal on Scientific Computing, 2023.

Parametric PDE

We consider PDEs parametrized by coefficent $a(x)$:

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), x \in D.$$

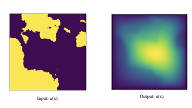


Motivation of Operator Learning: Meta-PINN?

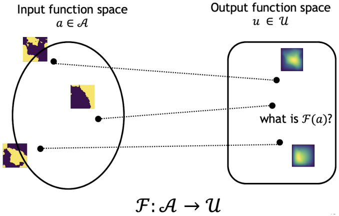
Parametric PDE

We consider PDEs parametrized by coefficent $a(x)$:

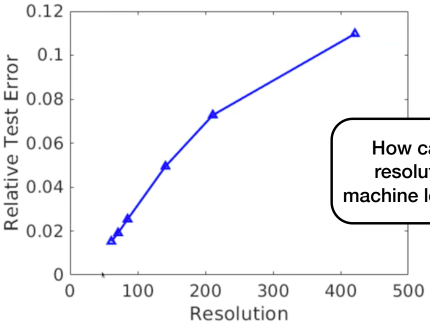
$$-\nabla \cdot (a(x)\nabla u(x)) = f(x), x \in D.$$



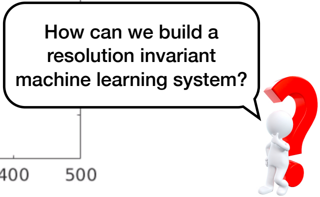
What if we have a dataset of $a(x)$



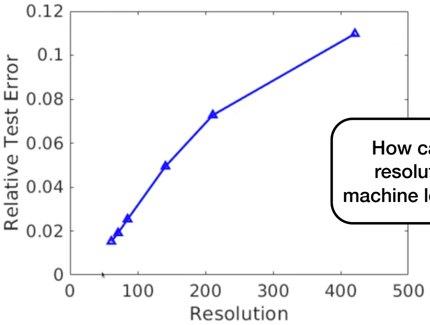
Khoo, Yuehaw, Jianfeng Lu, and Lexing Ying. "Solving parametric PDE problems with artificial neural networks." European Journal of Applied Mathematics

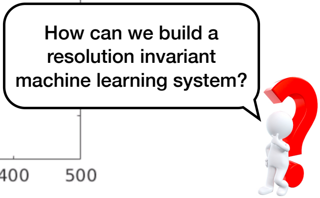


0.12
0.1
0.08
0.06
0.04
0.02
0
Relative Test Error
0
100
200
300
400
500
Resolution



How can we build a resolution invariant machine learning system?





Idea: Directly learn the mapping between functions.

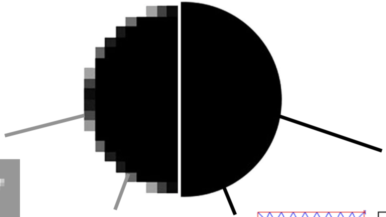
Lu L, Jin P, Karniadakis G E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:1910.03193, 2019.

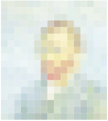
Kovachki N, Li Z, Liu B, et al. Neural operator: Learning maps between function spaces. arXiv preprint arXiv:2108.08481, 2021.

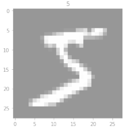
Operator Learning: General Framework

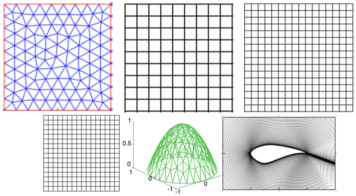
Operator Learning: Discretization-Invariant

Idea: Directly learn the mapping between functions.









Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

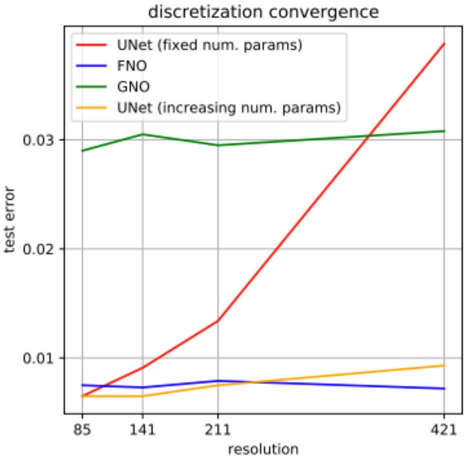
	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

Continuous function

Discretized vector

Idea: Learning in the function space



Operator Learning: Framework

Operator learning aims to build a parametric approximation $\mathcal{G}_\theta(\theta \in \mathbb{R}^p)$ to approximate a (non-linear) operator $\mathcal{G} : \underbrace{\mathcal{A}}_{\text{Banach space}} \to \underbrace{\mathcal{U}}_{\text{Banach space}}$

- Banach space $\mathcal{A} : \{a : D \to \mathbb{R}^{d_a}\}$ and $\mathcal{U} : \{a : D \to \mathbb{R}^{d_a}\}$ are all function space
- **Idea1:**

  Function → $\mathbb{R}^{d_1}$ Code → $\mathbb{R}^{d_2}$ Code → **Function**

  - Linear Encoding from a function to $\mathbb{R}^{d_1}$ code
  - Transform a $\mathbb{R}^{d_1}$ code to a $\mathbb{R}^{d_2}$ code
  - Linear Decoding from $\mathbb{R}^{d_2}$ code to a function

We need to generalize operations in neural networks to function space

We need to generalize operations in neural networks to function space

Linear Transform

- Linear Encoding: $u \to \{\int_x u(x) f_i(x) dx\}_{i=1}^n$

We need to generalize operations in neural networks to function space

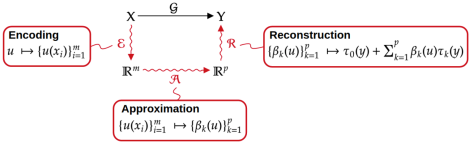
Linear Transform

- Linear Encoding: $u \to \{\int_x u(x) f_i(x) dx\}_{i=1}^n$
- a vector-input vector-output neural network

We need to generalize operations in neural networks to function space

Linear Transform

- Linear Encoding: $u \to \{\int_x u(x) f_i(x) dx\}_{i=1}^n$
- a vector-input vector-output neural network
- Linear Decoding: $\{\beta_k\}_{k=1}^p \to \sum_{k=1}^p \beta_k \underbrace{\tau_k}_{\text{function}}$



Universal approximation theorem of Chen & Chen (1995) states that DeepONets can approximate continuous operators

Lu L, Jin P, Pang G, et al. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature machine intelligence.

Operator Learning: Framework

Operator learning aims to build a parametric approximation $\mathcal{G}_\theta(\theta \in \mathbb{R}^p)$ to approximate a (non-linear) operator $\mathcal{G} : \underbrace{\mathcal{A}}_{\text{Banach space}} \to \underbrace{\mathcal{U}}_{\text{Banach space}}$

- Banach sapce $\mathcal{A} : \{a : D \to \mathbb{R}^{d_a}\}$ and $\mathcal{U} : \{a : D \to \mathbb{R}^{d_a}\}$ are all function space
- **Idea1**:

  Function $\longrightarrow$ $\mathbb{R}^{d_1}$ Code $\longrightarrow$ $\mathbb{R}^{d_2}$ Code $\longrightarrow$ **Function**

  - Linear Encoding from a function to $\mathbb{R}^{d_1}$ code
  - Transform a $\mathbb{R}^{d_1}$ code to a $\mathbb{R}^{d_2}$ code
  - Linear Decoding from $\mathbb{R}^{d_2}$ code to a function
- **Idea2**: Directly feature extraction in the function space!

We need to generalize operations in neural networks to function space

Convolution

- $$v_{l+1}(s) = \sigma\left(W_l v_l(s) + \underbrace{\int_D k_l(s,z) v_l(z) dz}_{\text{convolution}} + b_l s(s)\right)$$

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar A., "Neural Operator: Learning Maps Between Function Spaces", JMLR, 2021. doi:10.48550

We need to generalize operations in neural networks to function space

Convolution

- $v_{l+1}(s) = \sigma\left(W_l v_l(s) + \underbrace{\int_D k_l(s,z) v_l(z) dz}_{\text{convolution}} + b_l s(s) \right)$

  Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar A., "Neural Operator: Learning Maps Between Function Spaces", JMLR, 2021. doi:10.48550

- Fast implementation: Fourier Neural Operator

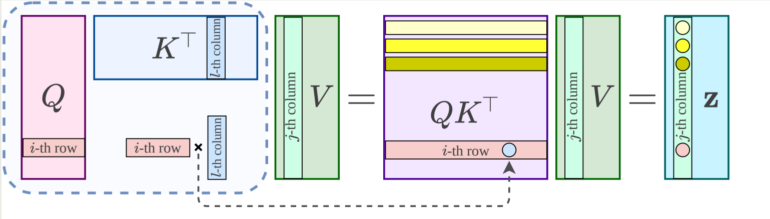
  FFT->multiplication->iFFT->nonlinear activation

  Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar A., "Fourier Neural Operator for Parametric Partial Differential Equations", ICLR, 2021.

We need to generalize operations in neural networks to function space

Attention

Original Attention: Fourier Transform $\int_\Omega (\xi_q(x_i)\phi_k(\xi))v_j(\xi)d\xi$



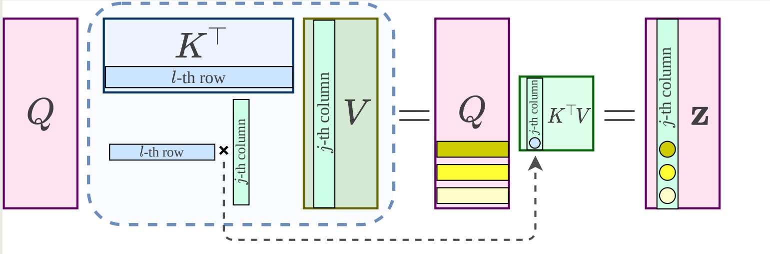
Computation scales $O(n^2k)$: n number of pixels, k number of ”Basis”

Cao S. Choose a transformer: Fourier or Galerkin. Advances in neural information processing systems, 2021, 34: 24924-24940.

We need to generalize operations in neural networks to function space

Attention

Galerkin Attention: $z_j(x_i) = \sum_{j=1}^{d} \left(\int_\Omega k_l(\xi) v_j(\xi) d\xi\right) q_l(x_i)$



Computation scales $O(nk^2)$: n number of pixels, k number of "Basis"

Cao S. Choose a transformer: Fourier or Galerkin. Advances in neural information processing systems, 2021, 34: 24924-24940.

Convergence Rate

- de Hoop M V, Nelsen N H, et al. Convergence rates for learning linear operators from noisy data. SIAM/ASA Journal on Uncertainty Quantification
- BoulleN, Townsend A. Learning elliptic partial differential equations with randomized linear algebra. Foundations of Computational Mathematics

Convergence Rate

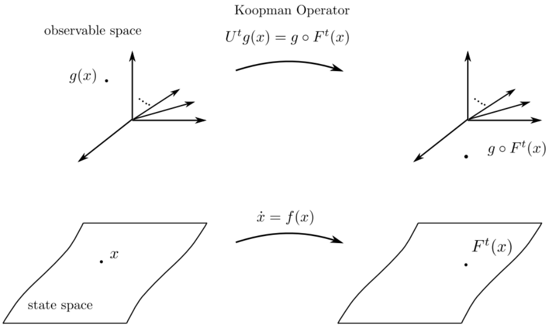
- de Hoop M V, Nelsen N H, et al. Convergence rates for learning linear operators from noisy data. SIAM/ASA Journal on Uncertainty Quantification
- BoulleN, Townsend A. Learning elliptic partial differential equations with randomized linear algebra. Foundations of Computational Mathematics

Improved Rates by Multi-level Methods

- Lin L, Lu J, Ying L. Fast construction of hierarchical matrix representation from matrix–vector multiplication. Journal of Computational Physics, 2011.
- BoulléN, Kim S, et al. Learning Green's functions associated with time-dependent partial differential equations. The Journal of Machine Learning Research.
- Schäfer F, Owhadi H. Sparse recovery of elliptic solvers from matrix-vector products. arXiv preprint arXiv:2110.05351, 2021.
- Jin J, Lu Y, Blanchet J, et al. Minimax Optimal Kernel Operator Learning via Multilevel Training, International Conference on Learning Representations. 2022.

Why Linear: Koopman Operator

The **Koopman** operator is a linear but infinite-dimensional operator that describes the evolution of observables in a finite dimensional dynamical system.



How the distribution of state space evolves through the dynamic!

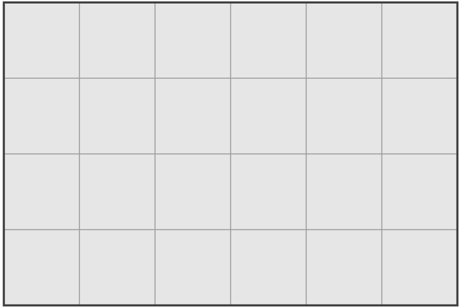
(Mathematically: adjoint of generator)

A linear operator is an "infinite-dimensional" matrix,

Hardness of learning in infinite dimensions: Linear Case

A linear operator is an "infinite-dimensional" matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.

Lin L, Lu J, Ying L. Fast construction of hierarchical matrix representation from matrix–vector multiplication. Journal of Computational Physics, 2011.



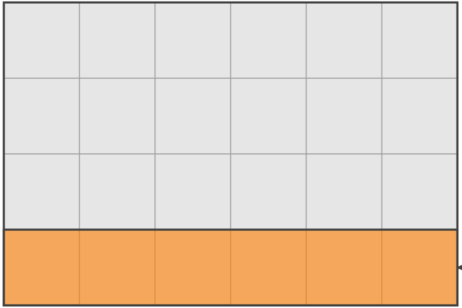
Multilevel algorithms are **essential** to achieve minimax optimality, which differs from finite-dimensional matrix reconstruction!

Jin J, Lu Y, Blanchet J, et al. Minimax Optimal Kernel Operator Learning via Multilevel Training, International Conference on Learning Representations. 2022.

Hardness of learning in infinite dimensions: Linear Case

A linear operator is an "infinite-dimensional" matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.

Lin L, Lu J, Ying L. Fast construction of hierarchical matrix representation from matrix–vector multiplication. Journal of Computational Physics, 2011.



Evert Row is a Linear Regression

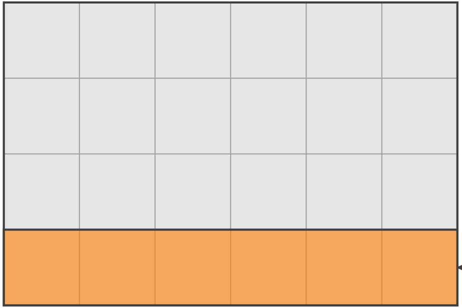
Multilevel algorithms are **essential** to achieve minimax optimality, which differs from finite-dimensional matrix reconstruction!

Jin J, Lu Y, Blanchet J, et al. Minimax Optimal Kernel Operator Learning via Multilevel Training, International Conference on Learning Representations. 2022.

Hardness of learning in infinite dimensions: Linear Case

A linear operator is an "infinite-dimensional" matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.

Lin L, Lu J, Ying L. Fast construction of hierarchical matrix representation from matrix–vector multiplication. Journal of Computational Physics, 2011.



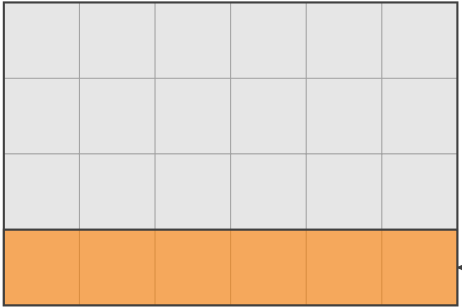
Optimal regularization differs for each row!

Evert Row is a Linear Regression

Hardness of learning in infinite dimensions: Linear Case

A linear operator is an "infinite-dimensional" matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.

Lin L, Lu J, Ying L. Fast construction of hierarchical matrix representation from matrix–vector multiplication. Journal of Computational Physics, 2011.



Optimal regularization differs for each row!

Evert Row is a Linear Regression

Multilevel algorithms are **essential** to achieve minimax optimality, which differs from finite-dimensional matrix reconstruction!

Jin J, Lu Y, Blanchet J, et al. Minimax Optimal Kernel Operator Learning via Multilevel Training, International Conference on Learning Representations. 2022.

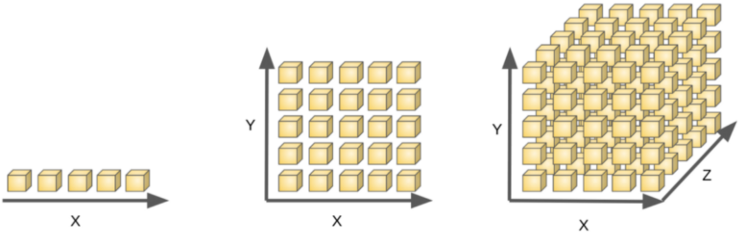
Neural operators can approximate any continuous operator. Chen & Chen 1995, Nikola Kovachki et. al. 2021

Hardness of learning in infinite dimensions

Neural operators can approximate any continuous operator. Chen & Chen 1995, Nikola Kovachki et. al. 2021

Curse of Dimensionality

The cost to represent a function is exponential to the dimensionality.



Smoothness only is not enough to break the Curse of dimensionality!

Different from finite dimension, **Smoothness** only is not enough to break the Curse of Dimensionty! Additional structure is needed, such as

- **Holomorphic Mappings**

  Schwab, C. & Zech, J. (2019), Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ, Analysis and Applications

- **PDE Oerators**

  Lanthaler S, Mishra S, Karniadakis G E. Error estimates for deeponets: A deep learning framework in infinite dimensions. Transactions of Mathematics and Its Applications, 2022, 6(1): tnac001.

Different from finite dimension, **Smoothness** only is not enough to break the Curse of Dimensionty! Additional structure is needed, such as

- **Holomorphic Mappings**

  Schwab, C. & Zech, J. (2019), Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ, Analysis and Applications

- **PDE Oerators**

  Lanthaler S, Mishra S, Karniadakis G E. Error estimates for deeponets: A deep learning framework in infinite dimensions. Transactions of Mathematics and Its Applications, 2022, 6(1): tnac001.

Open Question

What is the **general structure** that makes operator possible in infinite dimension?

Different structure that is used to break the curse of dimensionality includes

- Darcy Flow, Navier-Stokes via PCA-Net [1]
- Hamilton-Jacobi Equation [2]
- ...

Different structure that is used to break the curse of dimensionality includes

- Darcy Flow, Navier-Stokes via PCA-Net [1]
- Hamilton-Jacobi Equation [2]
- ...

Idea Neural Network can approximate known "algorithms"

- Approximate convergent schemes such as spectral methdos
- Approximate the Method of Characteristics

Similar to approximation theory for PINN

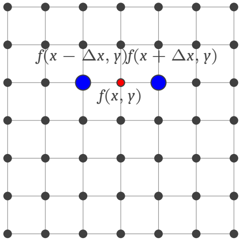
[1] Lanthaler S. Operator learning with PCA-Net: upper and lower complexity bounds. arXiv preprint arXiv:2303.16317, 2023.

[2] Lanthaler S, Stuart A M. The curse of dimensionality in operator learning. arXiv preprint arXiv:2306.15924, 2023.

Neural Operator adaptive to certain structure is ensential!

Approximate partial derivative using finite difference can be represent as convolution.

Approximate partial derivative using finite difference can be represent as convolution.

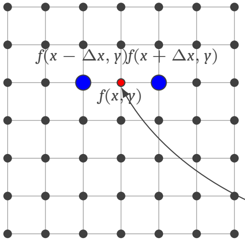


How to approximate $\partial_{xx} f(x, y)$?

Question

How can we map convolution kernels with finite difference operators?

Approximate partial derivative using finite difference can be represent as convolution.



How to approximate $\partial_{xx} f(x, y)$?

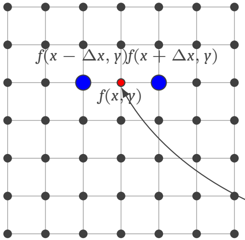
$$\partial_{xx} f(x, y) \approx \frac{f(x-\Delta x, y) + f(x+\Delta x, y) - 2f(x, y)}{\Delta x^2}$$

Question

How can we map convolution kernels with finite difference operators?

PDE-Net

Approximate partial derivative using finite difference can be represent as convolution.



How to approximate $\partial_{xx} f(x, y)$?

$$\partial_{xx} f(x, y) \approx \frac{f(x-\Delta x, y) + f(x+\Delta x, y) - 2f(x, y)}{\Delta x^2}$$

equivalent to conv kernel [-1,2,1]

Question

How can we map convolution kernels with finite difference operators?

What’s the property of a partial derivative?

Differentiation can because low order polynomial to zero!

What's the property of a partial derivative?

Differentiation can because low order polynomial to zero!

Orders of sum rules

For a filter q, we say q to have sum rules of order $\alpha = (\alpha_1, \alpha_2)$, where $\alpha \in \mathbb{Z}_+^2$, provided that

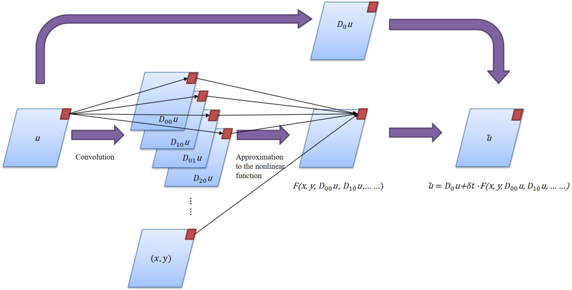
$$\sum_{k \in \mathbb{Z}^2} k^\beta q[k] = 0 \tag{16}$$

for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| < |\alpha|$ and for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| = |\alpha|$ but $\beta \neq \alpha$. If (16) holds for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| < K$ except for $\beta \neq \beta_0$ with certain $\beta_0 \in \mathbb{Z}_+^2$ and $|\beta_0| = J < K$, then we say q to have total sum rules of order $K \backslash \{J + 1\}$.

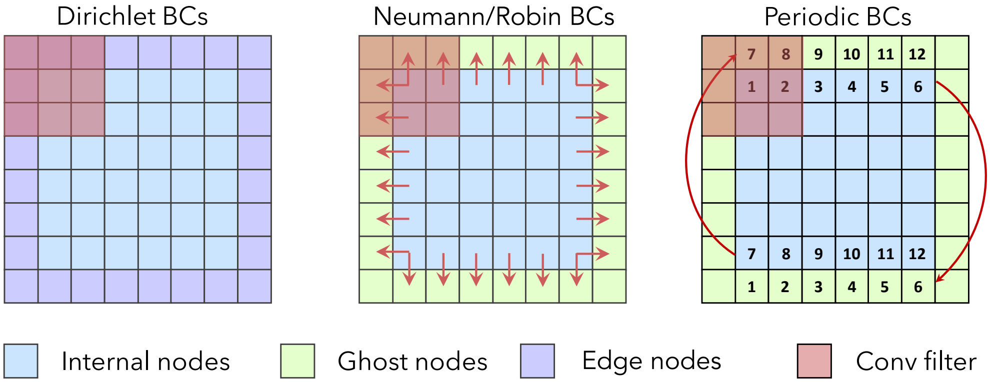
Linear constraints on convolutional weights!

PDE-Net is a neural network but can also represent a PDE with form

$$u_t = f(u, u_x, u_{xx}, \cdots)$$



- Linear constrained convolution kernel to approximate spatial derivatives
- 1x1 convolution kernel to approximate function f Lin M, Chen Q, Yan S. Network in network. arXiv preprint arXiv:1312.4400

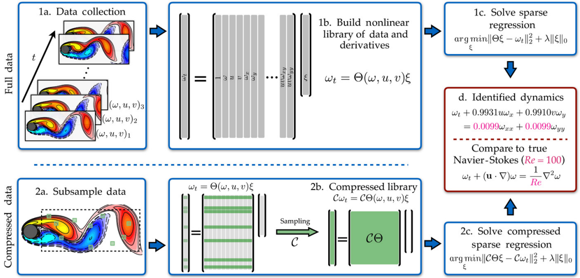


Rao C, Ren P, Wang Q, et al. Encoding physics to learn reaction–diffusion processes. Nature Machine Intelligence, 2023, 5(7): 765-779.

Build a big dictionary

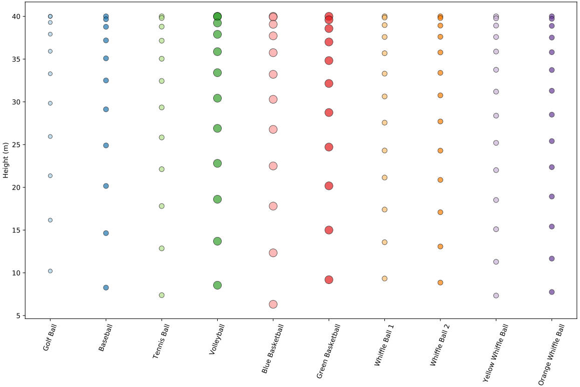
$$\Theta(U) = \underbrace{[1, U, U^2, \cdots, \sin(U), \cdots, U_x, U_x^2, \cdots, U_{xx}, U_{xx}^2, \cdots]}_{\text{possible dictionary}}$$

and then preform sparse regression methods



Brunton, Steven L.; Proctor, Joshua L.; Kutz, J. Nathan . "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". Proceedings of the National Academy of Sciences.

Chen Z, Liu Y, Sun H. Physics-informed learning of governing equations from scarce data. Nature communications.


Height (m)
40
35
30
25
20
15
10
5
Golf Ball
Baseball
Tennis Ball
Volleyball
Blue Basketball
Green Basketball
Whiffle Ball 1
Whiffle Ball 2
Yellow Whiffle Ball
Orange Whiffle Ball

Intuitively speaking, the balls in our data set (whiffle balls, perhaps, excluded) are similar enough objects that the equations governing their trajectories should include similar terms. Group Sparsity!

Ball	**First drop**	**Second drop**
Golf Ball	$\ddot{x} = -9.34 + 0.05v$	$\ddot{x} = -9.44 - 0.03v$
Baseball	$\ddot{x} = -8.51 + 0.14v$	$\ddot{x} = -7.56 + 0.14v$
Tennis Ball	$\ddot{x} = -9.08 - 0.13v$	$\ddot{x} = -8.64 - 0.12v$
Volleyball	$\ddot{x} = -8.11 - 0.08v$	$\ddot{x} = -9.64 - 0.23v$
Blue Basketball	$\ddot{x} = -6.71 + 0.15v$	$\ddot{x} = -7.50 + 0.07v$
Green Basketball	$\ddot{x} = -7.36 + 0.10v$	$\ddot{x} = -8.05 + 0.02v$
Whiffle Ball 1	$\ddot{x} = -8.24 - 0.34v$	$\ddot{x} = -9.44 - 0.43v$
Whiffle Ball 2	$\ddot{x} = -9.81 - 0.56v$	$\ddot{x} = -9.79 - 0.48v$
Yellow Whiffle Ball	$\ddot{x} = -8.50 - 0.47v$	$\ddot{x} = -8.45 - 0.46v$
Orange Whiffle Ball	$\ddot{x} = -7.83 - 0.35v$	$\ddot{x} = -8.03 - 0.42v$

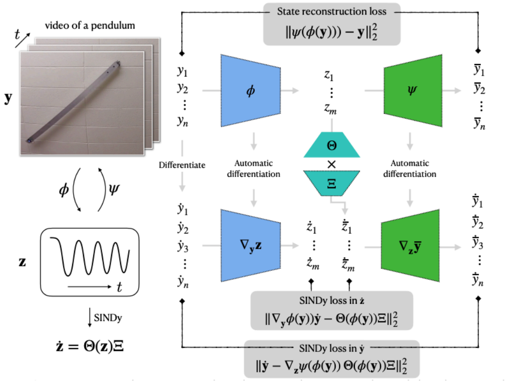
de Silva B M, Higdon D M, Brunton S L, et al. Discovery of physics from data: Universal laws and discrepancies. Frontiers in artificial intelligence, 2020, 3: 25.

The Reynolds number for a ball with diameter D and velocity v will then be

$$\text{Re} = 0.6667Dv \times 10^5$$

Ball	Radius (m)	Mass (kg)	Density (kg/m)	v_{max} (m/s)	Max Re
Golf Ball	0.021963	0.045359	1022.066427	26.63	1.75×10^5
Baseball	0.035412	0.141747	762.037525	26.61	2.83×10^5
Tennis Ball	0.033025	0.056699	375.813253	21.95	2.18×10^5
Volleyball	0.105*	NA	NA	22.09	6.96×10^5
Blue Basketball	0.119366	0.510291	71.628378	24.80	8.88×10^5
Green Basketball	0.116581	0.453592	68.342914	25.06	8.77×10^5
Whiffle Ball 1	0.036287	0.028349	141.641937	16.91	1.84×10^5
Whiffle Ball 2	0.036287	0.028349	141.641937	16.35	1.78×10^5
Yellow Whiffle Ball	0.046155	0.042524	103.250857	15.30	2.12×10^5
Orange Whiffle Ball	0.046155	0.042524	103.250857	15.77	2.18×10^5

Complex secondary physical mechanisms, like unsteady fluid drag forces, can obscure the underlying law of gravitation, leading to an erroneous model.



Bakarji J, Champion K, Nathan Kutz J, et al. Discovering governing equations from partial measurements with deep delay autoencoders. Proceedings of the Royal Society A, 2023, 479(2276): 20230422.

Different Levels of Interpretability

Fully white box, limited capacity

. . .

Gray box neural network

Physics knowledge as network structure, differentiable physics that integrate FEM/FDM solvers

. . .

Fully black box, universal approximator

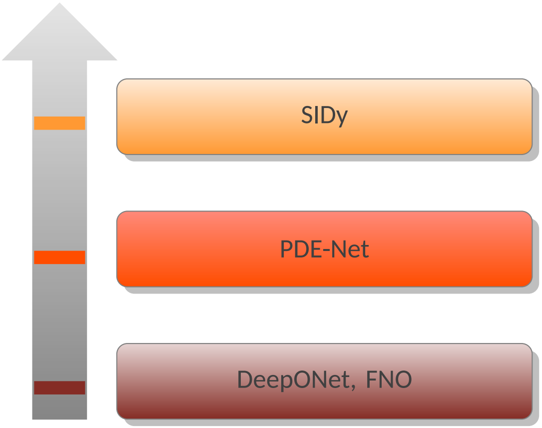
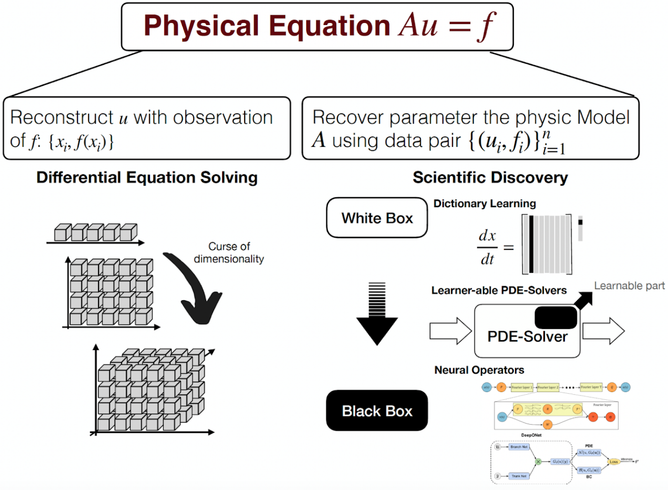


Table of Contents

Summary

In this tutorial, we introduced empirical and theoretical challenges to cooperate physical information $Au = f$ to machine learning systems



Recent Advances in Physics-Informed Machine Learning

Thank you for listening!

Any questions?