A Survey of Physics-informed Neural Networks and Non-differentiable Optimization

Yang Junyan

August 13, 2021

1 Introduction

Physics-informed Neural Networks (PINN) are neural networks that are trained to solve supervised learning tasks while respecting any given law of physics described by general nonlinear partial differential equations. It forms a new class of data-efficient universal function approximators that naturally encode any underlying physical laws as prior information.

2 Physics-informed Neural Networks

Here is the general form of the parameterized and nonlinear partial differential equations.

$$u_t + \mathcal{N}[u; \lambda] = 0, \tag{1}$$

where u(t, x) denotes the latent (hidden) solution and $\mathcal{N}[;\lambda]$ is a nonlinear differential operator parameterized by λ. one dimensional Burgers' equation as demo
two main classes of problem:
1. Given fixed model parameters λ what can be said about the unknown hidden state u(t, x) of the system?
2. Given the the observed data what are the parameters λ that best describe it?

2.1 Data-driven Solution

A major function of PINN is to obtain data driven solutions of partial differential equations, the general form of which is shown in equation 2.

$$u_t + \mathcal{N}[u] = 0, x \in \Omega, t \in [0, T], \tag{2}$$

where u(t, x) denotes the latent (hidden) solution, N [·] is a nonlinear differential operator, and Ω is a subset of $\mathbf{R}^D$. Considering the different nature of continuous-time and discrete-time problems, two corresponding sets of model were proposed.

2.1.1 Continuous Time Models

f(t,x) is defined as

$$f := u_t + \mathcal{N}[u], \tag{3}$$

then u(t,x) is approximated by a deep neural network. Such action result in a PINN f(t,x)

- Example: Burgers' Equation and Shrödinger Equation

2.1.2 Discrete Time Models

The general form of Runge-Kutta methods with q stages is expressed as

$$\begin{aligned} u^{n+c_i} &= u^n - \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}[u^{n+c_j}], i = 1, ..., q, \\ u^{n+1} &= u^n - \Delta t \sum_{j=1}^{q} b_j \mathcal{N}[u^{n+c_j}], \end{aligned} \tag{4}$$

where $u^{n+c_j}(x) = u(t^n + c_j, x)\ for\ j = 1, ..., q\ and\ a_{ij}, b_j, c_j$ are parameters. The equations above are equal to

$$\begin{aligned} u^n &= u_i^n,\ i = 1, ...q, \\ u^n &= u_{q+1}^n \end{aligned} \tag{5}$$

where

$$\begin{aligned} u_i^n &:= u^{n+c_i} + \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}[u^{n+c_j}], i = 1, ..., q, \\ u_{q+1}^n &:= u^{n+1} + \Delta t \sum_{j=1}^{q} b_j \mathcal{N}[u^{n+c_j}] \end{aligned} \tag{6}$$

A PINN that takes x as an input and $[u_1^n(x), ..., u_q^n(x), u_{q+1}^n(x)]$ as output can be constructed when neural network is placed prior on $[u^{n+c_1}(x), ..., u^{n+c_q}(x), u^{n+1}(x)]$.

- Example: Allen-Cahn Equation

2.2 Data-driven Discovery

Another major function of PINN is to perform data-driven discovery of partial differential equations.For example, in regard of equation 2, what are the parameters λ that best describe the observed data, which is a small set of scattered and potentially noisy observations of the hidden state u(t,x) of a system?

2.2.1 Continuous Time Models

The PINN f(t,x) is defined as

$$f := u_t + \mathcal{N}[u; \lambda], \tag{7}$$

where u(t,x) is approximated by a deep neural network.

- Example: Navier-Stokes Equation

2.2.2 Discrete Time Models

The general form of Runge-Kutta method is shown

$$\begin{aligned} u^{n+c_i} &= u^n - \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}[u^{n+c_j}; \lambda], i = 1, ..., q, \\ u^{n+1} &= u^n - \Delta t \sum_{j=1}^{q} b_j \mathcal{N}[u^{n+c_j}; \lambda], \end{aligned} \tag{8}$$

where $u^{n+c_j}(x) = u(t^n + c_j, x)\ for\ j = 1, ..., q\ and\ a_{ij}, b_j, c_j$ are parameters. The equation 8 can also be expressed as

$$\begin{aligned} u^n &= u_i^n,\ i = 1, ...q, \\ u^n &= u_i^{n+1},\ i = 1, ..., q. \end{aligned} \tag{9}$$

where

$$
\begin{aligned}
u_i^n &:= u^{n+c_i} + \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}[u^{n+c_j}; \lambda], i = 1, ..., q, \\
u_i^{n+1} &:= u^{n+c_i} + \Delta t \sum_{j=1}^{q} (a_{ij} - b_j) \mathcal{N}[u^{n+c_j}; \lambda], i = 1, ..., q.
\end{aligned}
\tag{10}
$$

By placing a neural network prior on $[u^{n+c_1}(x), ..., u^{n+c_q}(x)]$, we obtain two physics informed neural networks $[u_1^n(x), ..., u_q^n(x), u_{q+1}^n(x)]$ and $[u_1^{n+1}(x), ..., u_q^{n+1}(x), u_{q+1}^{n+1}(x)]$. Please be noted these two PINN share the same set of parameter as the causality between the input and output is identical. Therefore, with two sets of snapshots at t^n and t^{n+1}: x^n, u^n and x^{n+1}, u^{n+1}, we can train the PINN by minimizing the sum of squared errors

$$
SSE = SSE_n + SSE_{n+1} \tag{11}
$$

where

$$
\begin{aligned}
SSE_n &:= \sum_{j=1}^{q} \sum_{i=1}^{N_n} |u_j^n(x^{n,i}) - u^{n,i}|^2, \\
SSE_{n+1} &:= \sum_{j=1}^{q} \sum_{i=1}^{N_{n+1}} |u_j^{n+1}(x^{n+1,i}) - u^{n+1,i}|^2.
\end{aligned}
\tag{12}
$$

where $x^n = {x^{n,i}}_{i=1}^{N_n}$, $u^n = {u^{n,i}}_{i=1}^{N_n}$, $x^{n+1} = {x^{n+1,i}}_{i=1}^{N_{n+1}}$, and $u^{n+1} = {u^{n+1,i}}_{i=1}^{N_{n+1}}$

- Example: Korteweg–de Vries Equation

3 Non-differentiable Optimization

In progress...(Gradient Descent for current PINN, what's next)

4 Problem this week

- How do we define the loss?
  Here we take solving Schrodinger Equation as an example where a continuous time model is implemented.

Example (Shrödinger Equation)

This example aims to highlight the ability of our method to handle periodic boundary conditions, complex-valued solutions, as well as different types of nonlinearities in the governing partial differential equations. The nonlinear Schrödinger equation along with periodic boundary conditions is given by

$$\begin{aligned} &ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2], \\ &h(0, x) = 2\,\mathrm{sech}(x), \\ &h(t, -5) = h(t, 5), \\ &h_x(t, -5) = h_x(t, 5), \end{aligned}$$

where $h(t, x)$ is the complex-valued solution. Let us define $f(t, x)$ to be given by

$$f := ih_t + 0.5h_{xx} + |h|^2 h,$$

and proceed by placing a complex-valued neural network prior on $h(t, x)$. In fact, if u denotes the real part of h and v is the imaginary part, we are placing a multi-out neural network prior on $h(t, x) = \begin{bmatrix} u(t, x) & v(t, x) \end{bmatrix}$. This will result in the complex-valued (multi-output) physic informed neural network $f(t, x)$. The shared parameters of the neural networks $h(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss

$$MSE = MSE_0 + MSE_b + MSE_f,$$

where

$$MSE_0 = \frac{1}{N_0} \sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2,$$

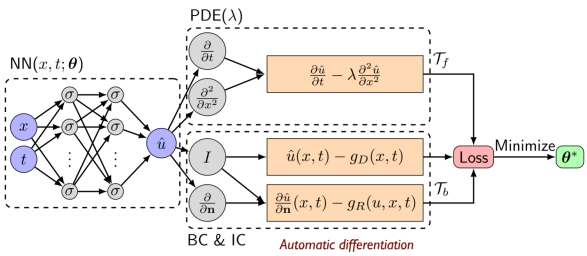
$$MSE_b = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(|h^i(t_b^i, -5) - h^i(t_b^i, 5)|^2 + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2 \right),$$

and

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Here, $\{x_0^i, h_0^i\}_{i=1}^{N_0}$ denotes the initial data, $\{t_b^i\}_{i=1}^{N_b}$ corresponds to the collocation points on the boundary, and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ represents the collocation points on $f(t, x)$. Consequently, MSE_0 corresponds to the loss on the initial data, MSE_b enforces the periodic boundary conditions, and MSE_f penalizes the Schrödinger equation not being satisfied on the collocation points.

Q: If there are other factors that may infect the loss besides loss(f), loss(u) and loss(b)?
P.S.: The above-mentioned three factors were listed in the CIS522 slides.



$$\mathcal{L}(\theta) := \underbrace{\mathcal{L}_u(\theta)}_{\text{Data fit}} + \underbrace{\mathcal{L}_r(\theta)}_{\text{PDE residual}} + \underbrace{\mathcal{L}_{u_0}(\theta)}_{\text{ICs fit}} + \underbrace{\mathcal{L}_{u_b}(\theta)}_{\text{BCs fit}}$$

- When is non-differential optimization needed?

References

[1] M.Raissi, P.Perdikaris, G.E.Karniadakis *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.* Journal of Computational Physics, 2019

[2] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis *Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations* . November 30, 2017