Physics Informed Neural Networks (PINNs)

物理法則に基づいた機械学習手法

YokoPhys-h

hoge 大学大学院

June 3, 2022

Contents

研究背景

例: 自由落下運動 [1]
運動方程式

$$m\frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2} = mg$$

1. 代数的解法
   微分方程式を解析的に解く.

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v(t) = gt + v_0$$

$$x_0 = \frac{1}{2}gt^2 + v_0t + x_0$$

非線形微分方程式など陽に解けない微分方程式がある.

2. シミュレーション的解法
微分方程式を Euler 法, Runge-Kutta 法などを用いて数値的に解く.

$$m\frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2} = mg \quad \Rightarrow \quad x(t+1) = x(t) = \Delta\frac{\mathrm{d}x(t)}{\mathrm{d}t}$$
$$\frac{\mathrm{d}x(t+1)}{\mathrm{d}t} = \frac{\mathrm{d}x(t)}{\mathrm{d}t} + \Delta g$$

数値解

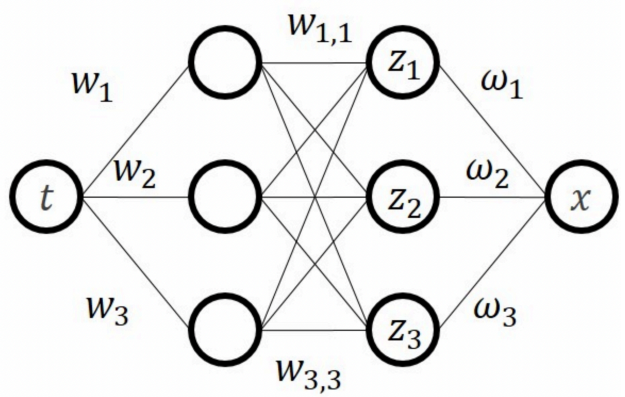
$$\begin{aligned} t &= 0 \quad x(0) = 0 \\ t &= 1 \quad x(1) = 4.9 \\ t &= 2 \quad x(2) = 19.6 \\ &\vdots \qquad\quad \vdots \end{aligned}$$

初期条件から順に計算する必要があり, 任意の時刻における変数値を求めることができない.

3. データ駆動アプローチ的解法
シミュレーションで求めた数値解を基に損失関数を定義し, 入出力の関係を Neural Networks で近似する.

数値解

$$t = 0 \quad x(0) = 0$$
$$t = 1 \quad x(1) = 4.9$$
$$t = 2 \quad x(2) = 19.6$$
$$\vdots \qquad \vdots$$



$$L = \sum_i ||x(t^i) - x_i||$$

Neural Network 解

$$x(t) = \omega_1 z_1 + \omega_2 z_2 + \omega_3 z_3$$
$$z_k(t) = h(w_{1,k} h(w_1 t) + w_{2,k} h(w_2 t) w_{3,k} h(w_3 t))$$

学習データが必要, 学習データの範囲内でのみ性能保証.

1. Physics Informed Neural Networks (PINNs)[1][2]

学習データの内容にとらわれずに Neural Networks を構築することはできないか?
Neural Networks 構築の上で学習データが大きく必要となる点は何か?
→ 損失関数

損失関数を学習データを用いず, 対象の方程式に沿った形で定義してやることができれば, 自然と対象の方程式の性質を孕んだ Neural Networks が構成できる.

背景

- 万能近似原理により, 深層学習が任意の関数を近似できる.
- 自動微分により, 出力変数の入力変数に対する全ての微分項が得られる.

[1] M. Raissi et al., *Journal of Computational Physics* **2019**, *378*, 686–707.
[2] M. Raissi et al., *Journal of Computational Physics* **2019**, *378*, 686–707.

PINNs の解決可能問題

- 微分方程式の求解
  微分方程式の初期条件及び境界条件を損失関数に用いることで偏微分方程式の解を求める.
- 微分方程式のパラメータ探索
  データをモデルとなる方程式に代入し, それと予測値の差から損失関数を構成し, パラメータを求める.

PINNs による物理量の求解

非線形微分方程式

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

$u(t, x)$: 潜在解 (latent (hidden) solution), $\mathcal{N}[\cdot; \lambda]$: 非線形項, $\omega \subset \mathbb{R}^D$

Chain-rule において自動微分を用いて Networks を構築する.
$\rightarrow$ 微分演算子 $\mathcal{N}$ の作用により活性化関数は異なるが、通常 $u(x, t)$ を表現する Network と同じだけのパラメータを持つ。

非線形微分方程式

$$f := u_t + \mathcal{N}[u] = 0, \quad x \in \Omega, t \in [0, T]$$

最小にすべき平均 2 乗誤差: $\mathrm{MSE} = \mathrm{MSE}_u + \mathrm{MSE}_f$

$$\mathrm{MSE}_u = \frac{1}{N_u} \sum_{i=1}^{N_u} \left| u\left(t_u^i, x_u^i\right) - u^i \right|^2$$

初期条件及び境界条件との差を評価

$$\mathrm{MSE}_f = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| f\left(t_f^i, x_f^i\right) \right|^2$$

f(x) を評価

例: 自由落下問題

$$MSE = \underbrace{||x(0) - x_0|| + \left|\left| \frac{\mathrm{d}x(0)}{\mathrm{d}t} - v_0 \right|\right|}_{\text{初期条件との損失評価}} + \underbrace{\sum_i \left|\left| m\frac{\mathrm{d}^2x(t^i)}{\mathrm{d}t^2} - mg \right|\right|}_{\text{微分方程式との損失評価}}$$

例: (1 次元非線形 Schrödinger 方程式)

$$f := i\psi_t + \frac{1}{2}\psi_{xx} + |\psi|^2\psi = 0, \quad x \in [-5, 5], t \in [0, \pi/2]$$

$\psi(0, x) = 2\,\mathrm{sech}(x)$: 初期条件, $\psi(t, -5) = \psi(t, 5), \psi_x(t, -5) = \psi_x(t, 5)$ 周期境界条件

(t : 時間, x : 位置, h : 複素数解)

最小にすべき平均 2 乗誤差: $\mathrm{MSE} = \mathrm{MSE}_u + \mathrm{MSE}_f$

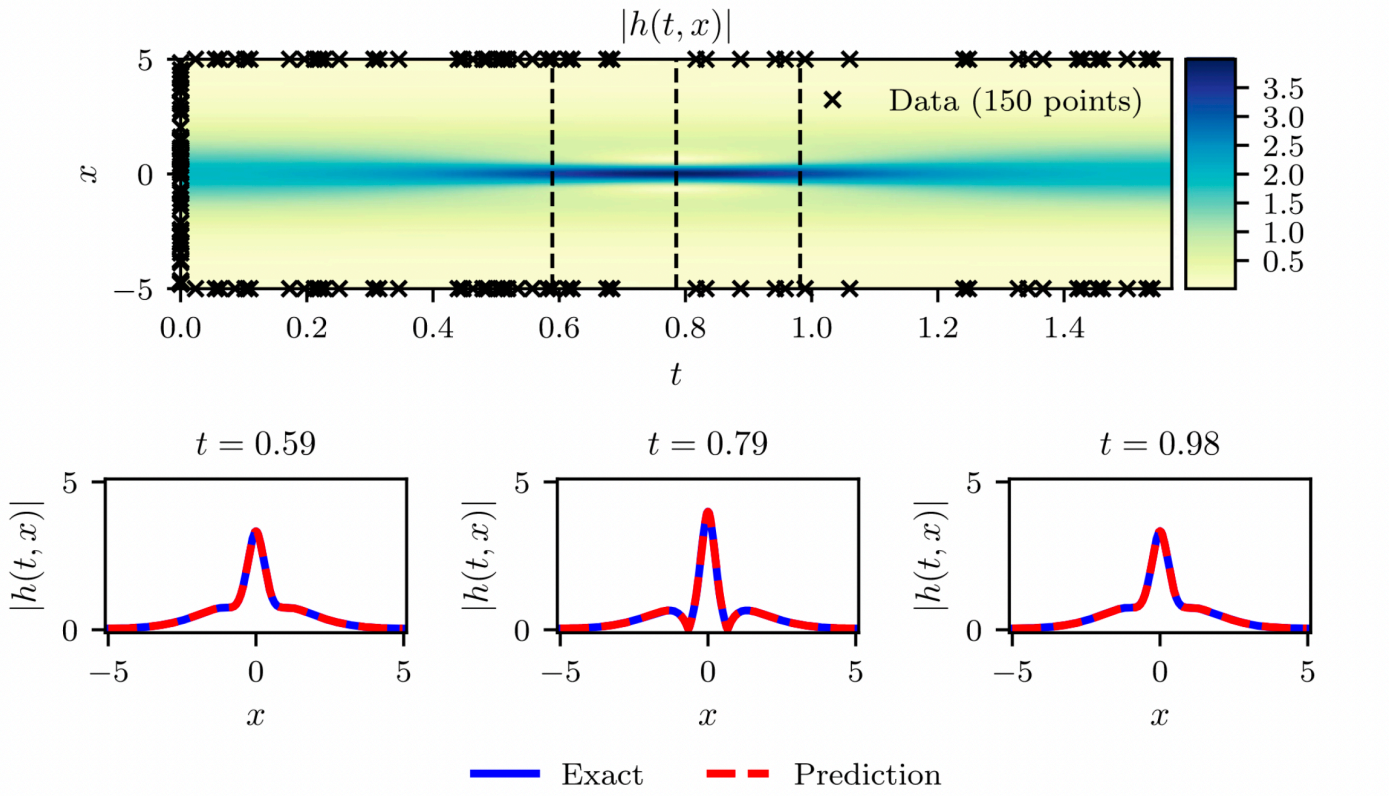
$$MSE_u = \frac{1}{N}\sum_{i=1}^{N}\left|u\left(t_u^i, x_u^i\right) - u^i\right|^2 \quad : \text{予測値の平均 2 乗誤差}$$

$$MSE_f = \frac{1}{N}\sum_{i=1}^{N}\left|f\left(t_u^i, x_u^i\right)\right|^2 \quad : \text{微分方程式の平均 2 乗誤差}$$

$$\mathrm{MSE} = |\hat{u} - u|^2 + |\hat{u}_t + \lambda_1\hat{u}\hat{u}_x - \lambda_2\hat{u}_{xx}|^2$$

Networksは複素数であるが, $\psi = u + iv$ としてそれぞれに Neural

- 計算結果



Data: 150 points, N_b: 50, N_f: 20000, 5-layer, 100 neurons

厳密なシミュレーションと比較して，その精度の高さを確認 $(\text{error} = 1.97 \times 10^3)$

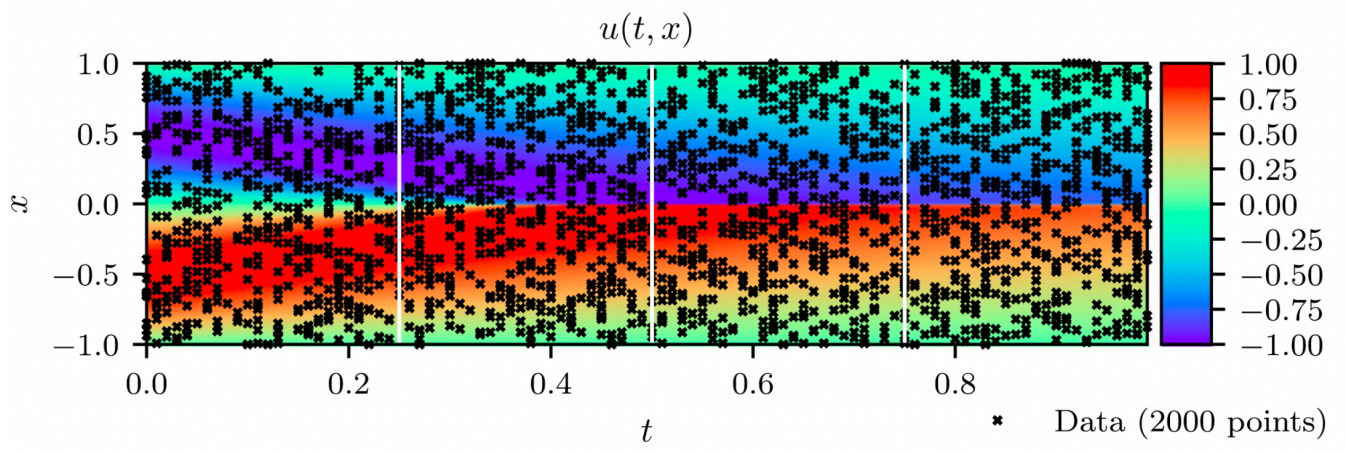
- 利点
  - 訓練データはランダムな初期データを与えただけ.
  - 少ないデータ量 (150 点) で再現可能.
- 問題点
  - 方程式に対する訓練データ N_f の多さ. 系の次元が増えるにつれ, 指数関数的に増加する.
    $\longrightarrow$ 時間発展に関する Runge-Kutta 法, Sparse Grid, quasi-Monte-Carlo sampling により解決

PINNs による微分方程式パラメータ探索

既に対象となるモデルがサンプリングできていて, かつそれが従う方程式がわかっている場合のパラメーター推定を行う.
例: Burgers 方程式

$$f := u_t + \lambda_1 - \lambda_2 u_{xx} = 0$$

学習データ: 既知の Burgers 方程式の厳密解の描像における, 適当なデータ点

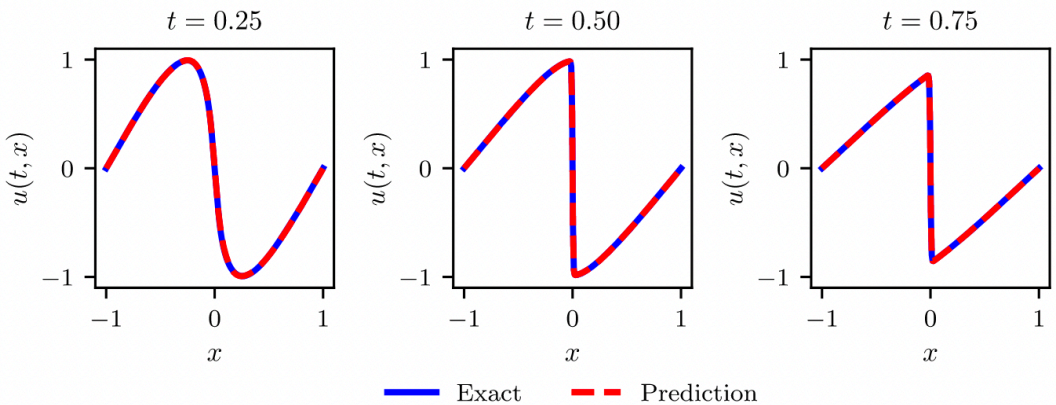


最小にすべき平均 2 乗誤差: $\mathrm{MSE} = \mathrm{MSE}_u + \mathrm{MSE}_f$

$$MSE_u = \frac{1}{N}\sum_{i=1}^{N}\left|u\left(t_u^i, x_u^i\right) - u^i\right|^2 \quad : \text{予測値の平均 2 乗誤差}$$

$$MSE_f = \frac{1}{N}\sum_{i=1}^{N}\left|f\left(t_u^i, x_u^i\right)\right|^2 \quad : \text{微分方程式の平均 2 乗誤差}$$

- 計算結果



Correct PDE	$u_t + uu_x - 0.0031831u_{xx} = 0$
Identified PDE (clean data)	$u_t + 0.99915uu_x - 0.0031794u_{xx} = 0$
Identified PDE (1% noise)	$u_t + 1.00042uu_x - 0.0032098u_{xx} = 0$

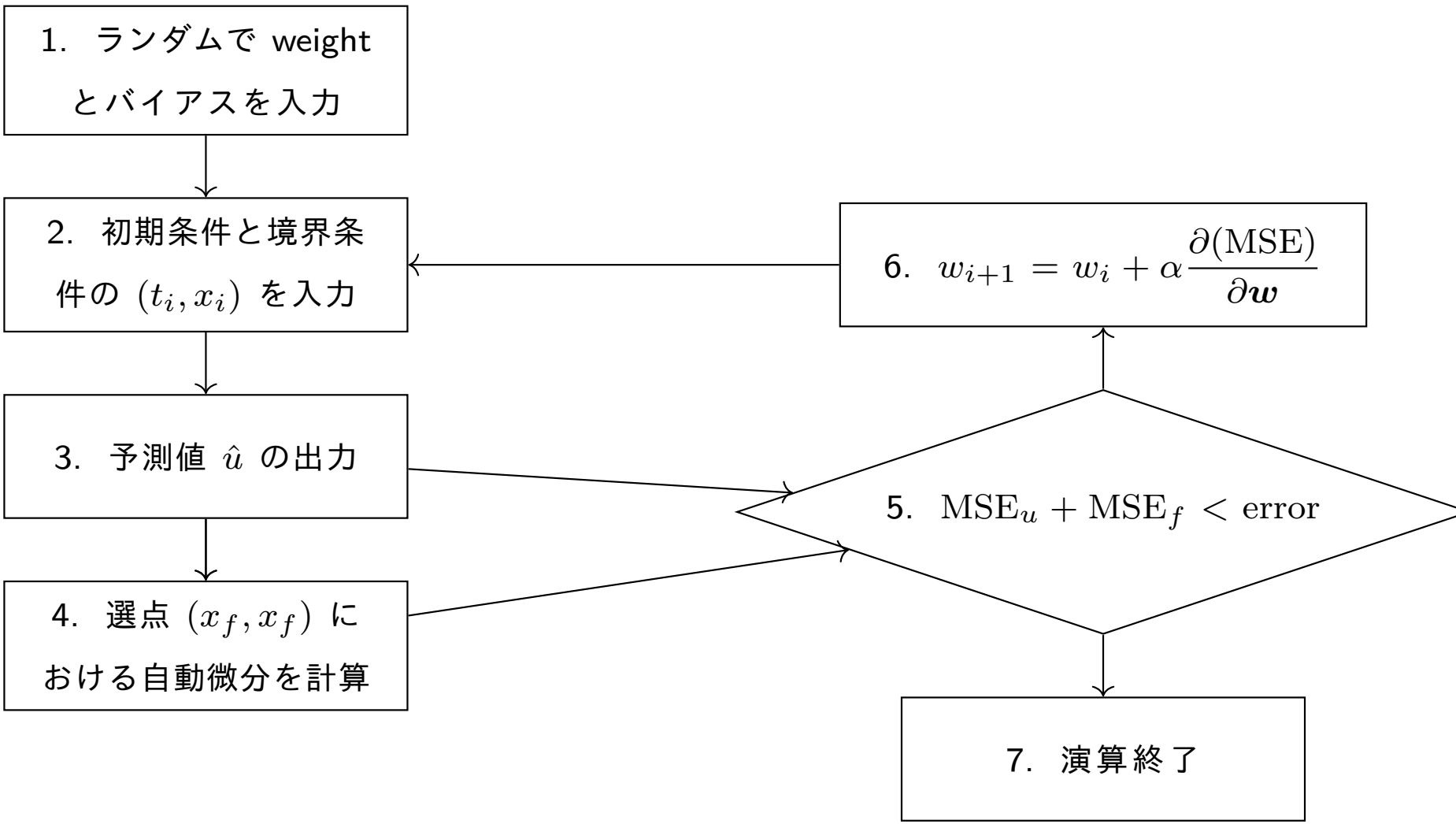
PINN の利点と問題点

- 利点
  - 損失関数の中に物理的要請を取り込むため, 学習データがランダム数でよい.
  - 学習データに依存しないため, 開発後の汎用性が高い.
  - 離散誤差を含むシミュレーション的手法より精度が高い.
- 問題点
  - 物理的要請 (初期条件, 境界条件) を損失関数として用いるため, そららが変わる場合, 再学習する必要がある.
    $\rightarrow$　境界条件が途中で変わる系とかキツい
  - 損失関数の定義にどこまで物理的要請を組み込めば良いかわからない.
  - 一応, 教師あり学習に当たるため, 事前に正解となるデータを蓄えておく必要がある.

今後個人的にすること

hgoehoge

計算フロー


1. ランダムで weight とバイアスを入力
2. 初期条件と境界条件の (t_i, x_i) を入力
3. 予測値 $\hat{u}$ の出力
4. 選点 (x_f, x_f) における自動微分を計算
5. $\mathrm{MSE}_u + \mathrm{MSE}_f < \mathrm{error}$
6. $w_{i+1} = w_i + \alpha \frac{\partial(\mathrm{MSE})}{\partial \boldsymbol{w}}$
7. 演算終了

自動微分

プログラムによって定義された数学的な関数が与えられた時に, その導関数をアルゴリズムによって機械的に求める. 例: $f(a,b) = ab + \sin b$ の導関数を機械的に計算する cite[hoge].

$$
\begin{aligned}
f &= w + u \\
w &= ab \\
u &= \sin b
\end{aligned}
\longrightarrow
\begin{aligned}
u' &= \cos b \cdot b' \\
w' &= a'b + ab' \\
f' &= w' + u'
\end{aligned}
\longrightarrow
\begin{aligned}
a' &= \frac{\partial a}{\partial a} = 1 \\
b' &= \frac{\partial b}{\partial b} = 0 \\
w' &= \cos b \cdot b' \\
f' &= w' + u' = b
\end{aligned}
$$

Chain-rule を繰り返し利用することで偏導関数値を少ない計算量で自動的に求めることができる. (ライブラリが豊富)

参考文献 I

[1] 話題の NVIDIA SimNet™ でも使われている Physics-Informed Neural Network について調べてみたら、深過ぎたので「自由落下」問題を Physics-Informed してみた。, https://qiita.com/nnnnnnn/items/df62e9fb0ec999df96a2.

[2] M. Raissi, P. Perdikaris, G. Karniadakis, *Journal of Computational Physics* **2019**, *378*, 686–707.

[3] M. Raissi, P. Perdikaris, G. E. Karniadakis, *Journal of Computational Physics* **2019**, *378*, 686–707.