Deep Learning methods for differential equations

薛名成 (Ming-Cheng Shiue)

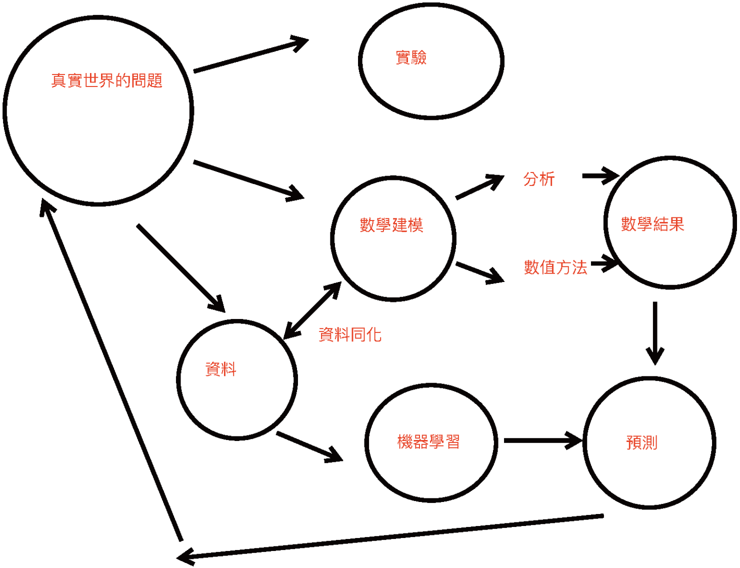
Department of Applied Mathematics,
National Yang Ming Chiao Tung University
(國立陽明交通大學應用數學系)

,

Seminar, October 18, 2023

Outlines

- Introduction to Scientific Computing
- Curse of dimensionality: solving Poisson equations in high dimensions based on finite difference
- Deep learning methods for Poisson equations: Physics-informed neural networks
- Summary


實驗
真實世界的問題
分析
數學建模
數學結果
數值方法
資料同化
資料
機器學習
預測

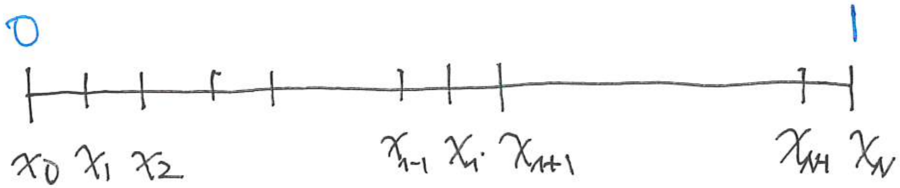
Finite Difference applied to solve Poisson equations

The model problem:
One-dimensional Poisson equations:

$$-\frac{\partial^2 u}{\partial x^2} = f(x), \quad x \in (0,1),$$

subject to boundary conditions

$$u(0) = u(1) = 0.$$



$\Delta x = \frac{1-0}{N}$, $\quad x_i = i\Delta x$, $\quad i = 0, 1, \cdots, N$

Approximate $u(x_i)$ by u_i, $\quad i = 0, \cdots, N$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)(x_i) \approx \frac{\left(\frac{\partial u}{\partial x}\right)_{i+\frac{1}{2}} - \left(\frac{\partial u}{\partial x}\right)_{i-\frac{1}{2}}}{\Delta x} \approx \frac{\frac{u_{i+1}-u_i}{\Delta x} - \frac{u_i - u_{i-1}}{\Delta x}}{\Delta x} = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}$$

$i = 1, 2, \cdots, N-1.$

Set $$-\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} = f_i, \quad i = 1, 2, \cdots, N-1, \quad f_i = f(x_i)$$

$u_0 = u_N = 0.$

$\hookrightarrow$ Difference scheme for 1-D Poisson equation

Set

$$U = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} \Rightarrow \text{linear system} \quad \underset{(N-1)\times(N-1)}{A} U = b$$

where

$$A = \frac{1}{(\Delta x)^2} \underset{(N-1)\times(N-1)}{\begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & -1 \\ 0 & & & -1 & 2 \end{pmatrix}}, \quad b = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{pmatrix}$$

Iterative algorithm for solving $AU=b$

Jacobi method: $A = D + R$ D: diagonal part of A

$$DU^{n+1} = b - RU^n \Rightarrow U^{n+1} = D^{-1}(b - RU^n)$$

Gauss-Seidel method: $A = L + \tilde{R}$ L: Lower triangular part of A

$$LU^{n+1} = b - \tilde{R}U^n \Rightarrow U^{n+1} = L^{-1}(b - \tilde{R}U^n).$$

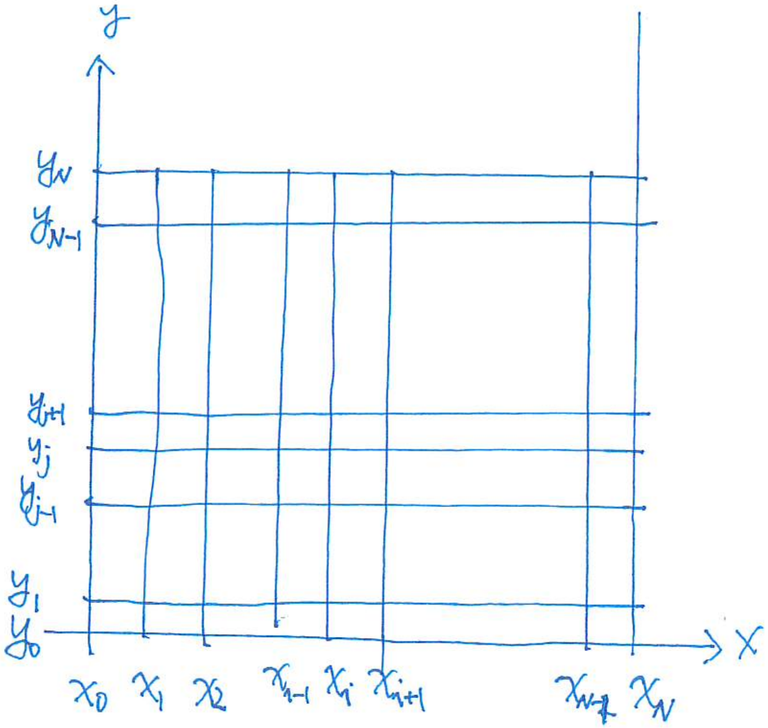
The model problem:
Two-dimensional Poisson equations:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x, y), \quad (x, y) \in (0, 1) \times (0, 1),$$

subject to boundary conditions

$$u(x, y) = 0, (x, y) \in \partial(0, 1) \times (0, 1).$$

2D Poisson equations



$$\Delta x = \Delta y = \frac{1-0}{N}$$

$$x_i = i\Delta x, \qquad i = 0, \cdots, N$$

$$y_j = j\Delta y, \qquad j = 0, \cdots, N$$

Approximate $u(x_i, y_j)$ by $u_{i,j}$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)(x_i, y_j) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2}$$

$$\left(\frac{\partial^2 u}{\partial y^2}\right)(x_i, y_j) \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2}$$

Numerical scheme:

$$-\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2}\right) = f_{i,j}, \qquad \begin{matrix} 1 \le i \le N-1 \\ 1 \le j \le N-1 \end{matrix}$$

$$u_{i,j} = 0 \qquad \text{if } i = 0 \text{ or } N,\ j = 0, \text{ or } N$$

Set

$$U = \begin{pmatrix} u_{1,1} \\ u_{1,2} \\ \vdots \\ u_{1,N-1} \\ u_{2,1} \\ \vdots \\ u_{2,N-1} \\ \vdots \\ u_{N-1,1} \\ \vdots \\ u_{N-1,N-1} \end{pmatrix}$$

linear system $\underset{(N-1)^2 \times (N-1)^2}{A}\, U = b$

where

$$A = \left(\begin{array}{cccc|cccc|cccc|c}
4 & -1 & & & -1 & & & & & & & & \\
-1 & 4 & -1 & & & -1 & & & & & & & \\
 & \ddots & \ddots & -1 & & & \ddots & & & & & & \\
 & & -1 & 4 & & & & -1 & & & & & \\
\hline
-1 & & & & 4 & -1 & & & -1 & & & & \\
 & -1 & & & -1 & 4 & -1 & & & \ddots & & & \\
 & & \ddots & & & \ddots & \ddots & -1 & & & \ddots & & \\
 & & & -1 & & & -1 & 4 & & & & -1 & \\
\hline
 & & & & -1 & & & & 4 & -1 & & & -1 \\
 & & & & & \ddots & & & -1 & 4 & -1 & & \ddots \\
 & & & & & & \ddots & & & \ddots & \ddots & -1 & \\
 & & & & & & & -1 & & & -1 & 4 & -1 \\
\hline
 & & & & & & \ddots & & & & \ddots & & \ddots
\end{array}\right), \quad b = \begin{pmatrix} f_{1,1} \\ f_{1,2} \\ \vdots \\ f_{1,N-1} \\ \vdots \\ f_{N-1,1} \\ \vdots \\ f_{N-1,N-1} \end{pmatrix}$$

Supervised learning

- Given the data pairs $\{(x_i, u_i = u(x_i))\}_{i=1}^{N}$ from the unknown solution $u(x)$
- Construct a finite family of neural network approximation $\{\Phi(x;\theta)\}_\theta$ θ: parameters
- Build some criteria to quantify how good $\Phi(x,\theta) \approx u(x)$

$$\text{Loss} = L(\theta) := \frac{1}{N}\sum_{i=1}^{N}(\Phi(x_i,\theta)-u_i)^2, \quad (\text{mean square loss})$$

- Find the best parameter θ by solving the optimization

$$\min_\theta L(\theta)$$

Deep learning-based PDEs solver

$$\begin{cases} -\Delta u = f & x \in \Omega \\ u|_{\partial\Omega} = 0 \end{cases}$$

- Construct a finite family of neural network approximation

$$\{\Phi(x;\theta)\}_\theta$$

- Build some criteria to quantify how good $\Phi(x,\theta)$?

$$-\Delta \Phi(x,\theta) \approx f \quad \text{and} \quad \Phi(x,\theta)|_{\partial\Omega} \overset{?}{\approx} 0$$

Loss function $\mathcal{L}(\theta)$

- Find the best parameter θ by solving the optimization

$$\min_\theta \mathcal{L}(\theta).$$

Neural Network Approximation

Let $d, L \in \mathbb{N}$. A neural network (NN) with input dimension d and L layers is a sequence of matrix-vector tuples

$$\Phi = ((A_1, b_1), (A_2, b_2), \cdots, (A_L, b_L))$$

where $N_0 = d$ and $N_1, \ldots, N_L \in \mathbb{N}$ and $A_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$, $b_\ell \in \mathbb{R}^{N_\ell}$, $\ell = 1, 2, \ldots, L$.

Given a NN Φ and an activation function $\rho : \mathbb{R} \to \mathbb{R}$,

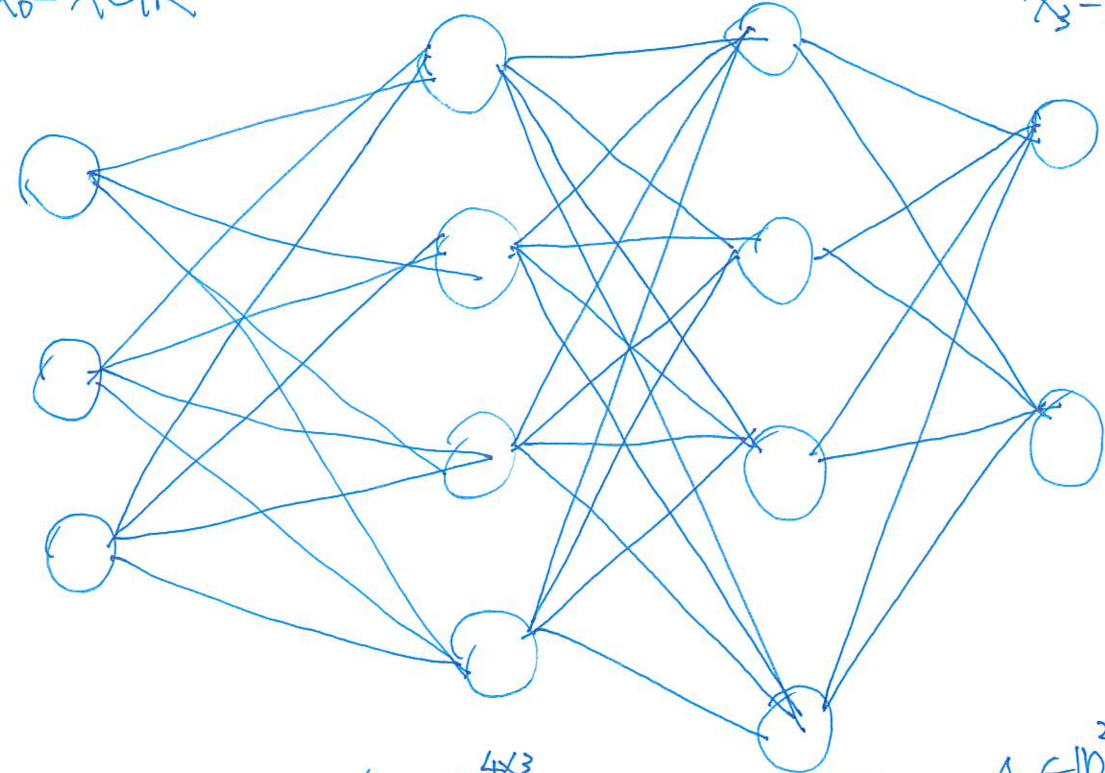
realization of Φ $\quad R(\Phi) : \mathbb{R}^d \longrightarrow \mathbb{R}^{N_L}$ by $x_L = R(\Phi)(x)$

$$x_0 = x$$
$$x_\ell = \rho(A_\ell x_{\ell-1} + b_\ell), \quad \ell = 1, \ldots, L-1,$$
$$x_L = A_L x_{L-1} + b_L$$

Example:

$x_2 = \rho(A_2 x_1 + b_2) \in \mathbb{R}^4$

$x_0 = x \in \mathbb{R}^3$ $\quad x_1 = \rho(A_1 x_0 + b_1) \in \mathbb{R}^4$ $\quad x_3 = A_3 x_2 + b_3 \in \mathbb{R}^2$



$A_1 \in \mathbb{R}^{4\times 3}$ $\quad A_2 \in \mathbb{R}^{4\times 4}$ $\quad A_3 \in \mathbb{R}^{2\times 4}$

$b_1 \in \mathbb{R}^4$ $\quad b_2 \in \mathbb{R}^4$ $\quad b_3 \in \mathbb{R}^2$

number of neurons

$$N(\Phi) = d + \sum_{j=1}^{L} N_j$$

number of layers

$$L(\Phi) = L$$

number of weights

$$M(\Phi) = \sum_{j=1}^{L} \left(\|A_j\|_0 + \|b_j\|_0 \right).$$

Central Question: (Universal Approximation Property)

Given function class $\mathcal{F}$ and $f \in \mathcal{F}$, what size should a neural network approximation $\hat{f}$ have such that

$$\hat{f} \approx f \, ?$$

Difficulty: generating training data can be expensive.

Physics Informed Neural Network learning method (PINN)

- select (grid) points $x_i \in \Omega$ and $y_j \in \partial\Omega$
- training sets $S_i = \{x_1, \cdots, x_N\} \subset \Omega$

$$S_b = \{y_1, \cdots y_M\} \subset \partial\Omega$$

- build loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \| -\Delta \Phi(x_i;\theta) - f(x_i) \|$

$$+ \frac{\lambda}{M} \sum_{j=1}^{M} \| \Phi(y_j;\theta) - 0 \|$$

λ: penalty parameter

- find the best parameter θ by choosing an optimization algorithm.
- SGD (stochastic gradient descent)
- Adam (Adaptive Moment Estimation)
- BFGS (Broyden-Fletcher-Goldfarb-Shanno)