Lecture 6: Solving PDEs via DNN Parametrization

Haizhao Yang
Department of Mathematics
University of Maryland College Park

2022 Summer Mini Course
Tianyuan Mathematical Center in Central China

Supervised Learning

- Given data pairs $\{(x_i, y_i = f(x_i))\}$ from an unknown map f;

Supervised Learning

- Given data pairs $\{(x_i, y_i = f(x_i))\}$ from an unknown map f;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;

Supervised Learning

- Given data pairs $\{(x_i, y_i = f(x_i))\}$ from an unknown map f;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;
- Create some criteria to quantify how good $\phi(x;\theta) \approx f(x)$ is:

$$\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\phi(x_i;\theta), y_i) \stackrel{\text{e.g.}}{=} \frac{1}{N}\sum_{i=1}^{N} (\phi(x_i;\theta) - y_i)^2;$$

Supervised Learning

- Given data pairs $\{(x_i, y_i = f(x_i))\}$ from an unknown map f;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;
- Create some criteria to quantify how good $\phi(x;\theta) \approx f(x)$ is:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\phi(x_i;\theta), y_i) \overset{\text{e.g.}}{=} \frac{1}{N}\sum_{i=1}^{N}(\phi(x_i;\theta) - y_i)^2;$$

- Use optimization to find the best θ;

Supervised Learning

- Given data pairs $\{(x_i, y_i = f(x_i))\}$ from an unknown map f;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;
- Create some criteria to quantify how good $\phi(x;\theta) \approx f(x)$ is:

$$\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\phi(x_i;\theta), y_i) \overset{\text{e.g.}}{=} \frac{1}{N}\sum_{i=1}^{N}(\phi(x_i;\theta) - y_i)^2;$$

- Use optimization to find the best θ;
- In a word: regression.

Deep Learning-Based PDE Solvers

- Given the PDE $\mathcal{D}u = f$ and boundary conditions $\mathcal{B}u = g$ with u unknown;

Deep Learning-Based PDE Solvers

- Given the PDE $\mathcal{D}u = f$ and boundary conditions $\mathcal{B}u = g$ with u unknown;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;

Deep Learning-Based PDE Solvers

- Given the PDE $\mathcal{D}u = f$ and boundary conditions $\mathcal{B}u = g$ with u unknown;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;
- Create some criteria to quantify how good $\mathcal{D}\phi(x;\theta) \approx f(x)$ and $\mathcal{B}\phi(x;\theta) \approx g(x)$ are;

Deep Learning-Based PDE Solvers

- Given the PDE $\mathcal{D}u = f$ and boundary conditions $\mathcal{B}u = g$ with u unknown;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;
- Create some criteria to quantify how good $\mathcal{D}\phi(x;\theta) \approx f(x)$ and $\mathcal{B}\phi(x;\theta) \approx g(x)$ are;
- Use optimization to find the best θ;

Deep Learning-Based PDE Solvers

- Given the PDE $\mathcal{D}u = f$ and boundary conditions $\mathcal{B}u = g$ with u unknown;
- Construct a finite family of maps $\{\phi(x;\theta)\}_\theta$;
- Create some criteria to quantify how good $\mathcal{D}\phi(x;\theta) \approx f(x)$ and $\mathcal{B}\phi(x;\theta) \approx g(x)$ are;
- Use optimization to find the best θ;
- In a word: regression.

Different criteria lead to different methods

- Least square methods (DGM, PINN);
- Variational methods (Deep Ritz);
- Adversarial methods (WAN, Select-Net, Friedrich learning);

Different optimization lead to different methods

- SGD with a fixed DNN (most methods);
- Growing width (Xu et al. and Cai et al.)
- Growing depth (Hao et al.)

Boundary Value Problem (BVP)

Given a PDE problem,

$$\mathcal{D}(u) = f \quad \text{in } \Omega,$$
$$\mathcal{B}(u) = g \quad \text{on } \partial\Omega.$$

A DNN $\phi(\boldsymbol{x};\boldsymbol{\theta}^*)$ is constructed to approximate the solution $u(\boldsymbol{x})$ via

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \\
&:= \arg\min_{\boldsymbol{\theta}} \|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\|_2^2 + \lambda\|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})\|_2^2 \\
&= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2\right] + \lambda\mathbb{E}_{\boldsymbol{x}\in\partial\Omega}\left[|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})|^2\right] \\
&\approx \arg\min_{\boldsymbol{\theta}} \frac{1}{N_1}\sum_{i=1}^{N_1}|\mathcal{D}\phi(\boldsymbol{x}_i;\boldsymbol{\theta}) - f(\boldsymbol{x}_i)|^2 + \lambda\frac{1}{N_2}\sum_{j=1}^{N_2}|\mathcal{B}\phi(\boldsymbol{x}_j;\boldsymbol{\theta}) - g(\boldsymbol{x}_j)|^2
\end{aligned}$$

with random samples $\boldsymbol{x}_i$ in Ω and $\boldsymbol{x}_j$ on $\partial\Omega$.

Main idea: find ϕ such that it can fit the “label" f after $\mathcal{D}$ at randomly sampled arbitrary sample locations.

Least Square Methods

Problem

$$\begin{aligned} \mathcal{D}(u) &= f \quad \text{in } \Omega, \\ \mathcal{B}(u) &= g \quad \text{on } \partial\Omega. \end{aligned}$$

Stochastic discrete method

- Randomly generate sample sets Ω^r and $\partial\Omega^r$
- Define a random loss function

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \Omega^r, \partial\Omega^r) \quad := \quad & \frac{1}{|\Omega^r|} \sum_{\boldsymbol{x} \in \Omega^r} \left[|\mathcal{D}\phi(\boldsymbol{x}; \boldsymbol{\theta}) - f(\boldsymbol{x})|^2 \right] \\ & + \frac{\lambda}{|\partial\Omega^r|} \sum_{\boldsymbol{x} \in \partial\Omega^r} \left[|\mathcal{B}\phi(\boldsymbol{x}; \boldsymbol{\theta}) - g(\boldsymbol{x})|^2 \right]. \end{aligned}$$

- Update via gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\partial \mathcal{L}(\boldsymbol{\theta}, \Omega^r, \partial\Omega^r)}{\partial \boldsymbol{\theta}}$$

Least Square Methods

Stochastic gradient descent method

- Randomly generate sample sets Ω^r and $\partial\Omega^r$
- Define a random loss function

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \Omega^r, \partial\Omega^r) \quad := \quad & \frac{1}{|\Omega^r|}\sum_{\boldsymbol{x}\in\Omega^r}\left[\left|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\right|^2\right] \\ & + \frac{\lambda}{|\partial\Omega^r|}\sum_{\boldsymbol{x}\in\partial\Omega^r}\left[\left|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})\right|^2\right].\end{aligned}$$

- Update via gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\frac{\partial\mathcal{L}(\boldsymbol{\theta}, \Omega^r, \partial\Omega^r)}{\partial\boldsymbol{\theta}}$$

Questions

- Convergence guarantee? Only partially known
- Fast convergence? Not available
- How good local minimizers are? Not known

Least Square Methods

Problem

$$\mathcal{D}(u) = f \quad \text{in } \Omega,$$
$$\mathcal{B}(u) = g \quad \text{on } \partial\Omega.$$

Continuous method

A DNN $\phi(\boldsymbol{x};\boldsymbol{\theta}^*)$ is constructed to approximate the solution $u(\boldsymbol{x})$ via

$$\begin{aligned} \boldsymbol{\theta}^* &= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2\right] + \lambda \mathbb{E}_{\boldsymbol{x}\in\partial\Omega}\left[|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})|^2\right]. \end{aligned}$$

Observation

- Non-uniqueness of network representation.
- Denseness of good local minimizers.
- Fast convergence to good approximate solutions.

How to use this deep learning-based PDE solver?

To be answered later.

Boundary Conditions and Network Design

Problem

$$\mathcal{D}(u) = f \quad \text{in } \Omega,$$
$$\mathcal{B}(u) = g \quad \text{on } \partial\Omega.$$

A naive method

A DNN $\phi(\boldsymbol{x};\boldsymbol{\theta}^*)$ is constructed to approximate the solution $u(\boldsymbol{x})$ via

$$\begin{aligned}\boldsymbol{\theta}^* &= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[\left|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\right|^2\right] + \lambda\mathbb{E}_{\boldsymbol{x}\in\partial\Omega}\left[\left|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})\right|^2\right].\end{aligned}$$

Not easy to solve the soft-constrained optimization

Boundary Conditions and Network Design

Problem

$$\mathcal{D}(u) = f \quad \text{in } \Omega,$$
$$\mathcal{B}(u) = g \quad \text{on } \partial\Omega.$$

A naive method

A DNN $\phi(\boldsymbol{x};\boldsymbol{\theta}^*)$ is constructed to approximate the solution $u(\boldsymbol{x})$ via

$$\begin{aligned}\boldsymbol{\theta}^* &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[\left|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\right|^2\right] + \lambda\mathbb{E}_{\boldsymbol{x}\in\partial\Omega}\left[\left|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})\right|^2\right].\end{aligned}$$

Not easy to solve the soft-constrained optimization

Possible solution

- Construct networks satisfy the boundary conditions automatically
- Reduce the soft-constrained optimization to

$$\begin{aligned}\boldsymbol{\theta}^* &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[\left|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\right|^2\right]\end{aligned}$$

Boundary Conditions and Network Design

Dirichlet boundary condition

- Assume $u(a) = a_0$, $u(b) = b_0$ for simplicity
- Introduce $h_1(x)$ and $l_1(x)$ to augment $\hat{u}(x;\boldsymbol{\theta})$ to obtain the final network $u(x;\boldsymbol{\theta})$:

$$u(x;\boldsymbol{\theta}) = h_1(x)\hat{u}(x;\boldsymbol{\theta}) + l_1(x).$$

- $l_1(x)$ satisfies the given Dirichlet boundary condition, i.e. $l_1(a) = a_0$, $l_1(b) = b_0$
- $h_1(x)$ satisfies the homogeneous Dirichlet boundary condition, i.e. $h_1(a) = 0$, $h_1(b) = 0$
- e.g.,

$$l_1(x) = (b_0 - a_0)(x - a)/(b - a) + a_0.$$

  and

$$h_1(x) = (x - a)^{p_a}(x - b)^{p_b},$$

  with $0 < p_a,\ p_b \leq 1$.

Boundary Conditions and Network Design

One-sided boundary condition

- Assume $u(a) = a_0$, $u'(a) = a_1$ for simplicity
- The final network $u(x;\boldsymbol{\theta}) = h_2(x)\hat{u}(x;\boldsymbol{\theta}) + l_2(x)$
- e.g.,

  $$l_2(x) = a_1(x - a) + a_0,$$

  and

  $$h_2(x) = (x - a)^{p_a},$$

  with $1 < p_a \leq 2$

Boundary Conditions and Network Design

Mixed boundary condition

- Assume $u'(a) = a_0$, $u(b) = b_0$ for simplicity
- The final network

$$u(x;\boldsymbol{\theta}) = (x-a)^{p_a}\hat{u}(x;\boldsymbol{\theta}) - (b-a)^{p_a}\hat{u}(b;\boldsymbol{\theta}) + l_3(x).$$

- e.g.,

$$l_3(x) = a_0 x + b_0 - a_0 b$$

Boundary Conditions and Network Design

Neumann boundary condition

- Assume $u'(a) = a_0$, $u'(b) = b_0$
- $$u(x;\boldsymbol{\theta}) = \exp(\frac{p_a x}{a-b})(x-a)^{p_a}\big((x-b)^{p_b}\hat{u}(x;\hat{\boldsymbol{\theta}})+c_2\big)+c_1+l_4(x), \quad (1)$$
  where $\boldsymbol{\theta} = \{\hat{\boldsymbol{\theta}}, c_1, c_2\}$ and
  $$l_4(x) = \frac{(b_0 - a_0)}{2(b-a)}(x-a)^2 + a_0 x.$$

Other PDE Problems

Typical PDE problems of interest can be summerized as:

- Eigenvalue problem:

$$\begin{aligned} \mathcal{D}u(\boldsymbol{x}) &= \lambda u(\boldsymbol{x}) \text{ in } \Omega, \\ \mathcal{B}u(\boldsymbol{x}) &= g_0(\boldsymbol{x}) \text{ on } \partial\Omega. \end{aligned} \tag{2}$$

- Parabolic equation:

$$\begin{aligned} &\frac{\partial u(\boldsymbol{x},t)}{\partial t} - \mathcal{D}u(\boldsymbol{x},t) = f(\boldsymbol{x},t) \text{ in } \Omega \times (0,T), \\ &\mathcal{B}u(\boldsymbol{x},t) = g_0(\boldsymbol{x},t) \text{ on } \partial\Omega \times (0,T), \\ &u(\boldsymbol{x},0) = h_0(\boldsymbol{x}) \text{ in } \Omega. \end{aligned} \tag{3}$$

- Hyperbolic equation:

$$\begin{aligned} &\frac{\partial^2 u(\boldsymbol{x},t)}{\partial t^2} - \mathcal{D}u(\boldsymbol{x},t) = f(\boldsymbol{x},t) \text{ in } \Omega \times (0,T), \\ &\mathcal{B}u(\boldsymbol{x},t) = g_0(\boldsymbol{x},t) \text{ on } \partial\Omega \times (0,T), \\ &u(\boldsymbol{x},0) = h_0(\boldsymbol{x}), \quad \frac{\partial u(\boldsymbol{x},0)}{\partial t} = h_1(\boldsymbol{x}) \text{ in } \Omega. \end{aligned} \tag{4}$$

These problems can be treated as BVP when t is considered as a spatial variable

Variation Methods

Example: Deep Ritz (E and Yu)

- Consider an example of an elliptic PDE with a homogeneous Dirichlet boundary condition

$$-\Delta u(\boldsymbol{x}) + c(\boldsymbol{x})u(\boldsymbol{x}) = f(\boldsymbol{x}),\ \boldsymbol{x} \in \Omega, \qquad u(\boldsymbol{x}) = 0,\ \boldsymbol{x} \in \partial\Omega,$$

  where c is a bounded function and $f \in L^2$.
- Then the solution u minimizes a variation formulation

$$\frac{1}{2}\int_\Omega \|\nabla u\|^2 + cu^2 \mathrm{d}\boldsymbol{x} - \int_\Omega fu \mathrm{d}\boldsymbol{x}.$$

- Final loss function with a penalty term for boundary

$$\mathcal{L}(u) := \frac{1}{2}\int_\Omega \|\nabla u\|^2 + cu^2 \mathrm{d}\boldsymbol{x} - \int_\Omega fu \mathrm{d}\boldsymbol{x} + \lambda \int_{\partial\Omega} u^2 \mathrm{d}\boldsymbol{x}.$$

Example 1: Weak Adversarial Method

- Consider the same PDE in Deep Ritz
- Let $v \in H_0^1(\Omega)$ be a test function
- The weak solution u is defined as the function that satisfies the bilinear equations:

$$\begin{aligned} a(u,v) :&= \int_\Omega \nabla u \nabla v + cuv - fv \mathrm{d}\boldsymbol{x} = 0, \quad \forall v \in H_0^1(\Omega), \\ u(\boldsymbol{x}) &= 0, \ \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{5}$$

- Min-max formulation:

$$\min_{u \in H_0^1(\Omega)} \max_{v \in H_0^1(\Omega)} |a(u,v)|^2 / \|v\|^2_{L^2(\Omega)}. \tag{6}$$

- Final loss functional $\mathcal{L}$ to identify the PDE solution as

$$\mathcal{L}(u) := \max_{v \in H_0^1(\Omega)} |a(u,v)|^2 / \|v\|^2_{L^2(\Omega)} + \lambda \int_{\partial\Omega} u^2 \mathrm{d}\boldsymbol{x}. \tag{7}$$

How to use deep learning-based PDE solvers?

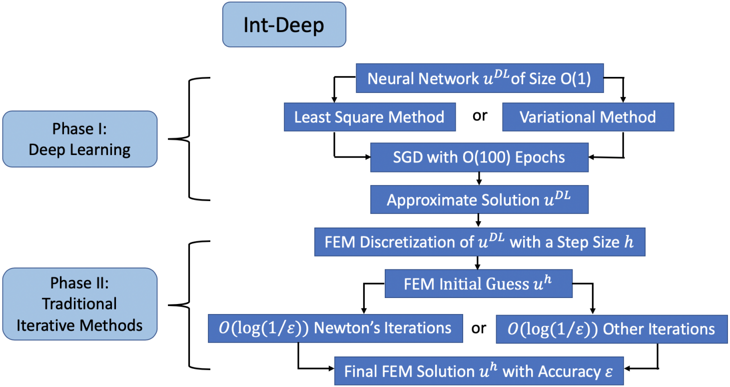
High dimensional PDEs

- Idea: probably no curse of dimensionality
- Practice: accuracy becomes poor when dimension increases
- Bottleneck: MC method for high dimensional integral

How to use deep learning-based PDE solvers?

Low dimensional PDEs.

- Idea: DL for initial guesses in traditional iterative methods
- Int-Deep: A Deep Learning Initialized Iterative Method for Nonlinear Problems (Huang et al.)



Different criteria lead to different methods

- Least square methods (DGM, PINN);
- Variational methods (Deep Ritz);
- Adversarial methods (WAN, Select-Net, Friedrich learning);

Different optimization lead to different methods

- SGD with a fixed DNN (most methods);
- Growing width (Xu et al. and Cai et al.)
- Growing depth (Hao et al.)