Weak Adversarial Network (WAN): A Deep Learning Method for Forward and Inverse Problems with High Dimensional PDEs

Haomin Zhou

School of Mathematics, Georgia Tech

Joint work with Gang Bao (Zhejiang), Xiaojing Ye (Georgia State), and Yaohua Zang (Zhejiang)

Partially supported by NSF

Outline

Motivation

Goal: numerically solve forward and inverse problems with PDEs in high dimensions.

example: elliptic equation in $\Omega \in \mathbb{R}^d$ (arbitrary shape),

$$\begin{cases} -\sum_{i=1}^{d} \partial_i(a_{ij}\partial_j u) = f, & \text{in } \Omega \\ u(x) - g(x) = 0 \quad \text{(Dirichlet)} & \text{on } \partial\Omega \end{cases}$$

Challenge: The computational cost for conventional methods (Finite Difference, Finite Elements, Spectral, and others) becomes intractable when the dimension is high.

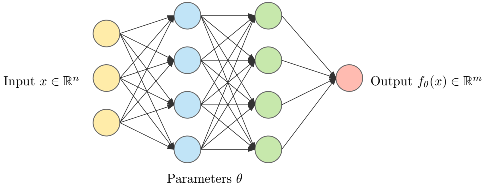
Our Strategies: Leveraging a minimax framework (2-player game strategy) and neural networks.

Neural Networks

Deep neural networks are compositions of multiple simple functions (called layers) so that they can approximate complicated functions. For example:

$$f_\theta(x) = w_K^T l_{K-1} \circ \cdots \circ l_0(x) + b_K,$$

where k-th layer $l_k(z) = \sigma_k(W_k z + b_k)$ with weight $W_k \in \mathbb{R}^{d_{k+1} \times d_k}$ and bias $b_k \in \mathbb{R}^{d_{k+1}}$. Parameters θ are collections of (w_K, b_k, W_k).



Training of neural networks: find the best parameters to minimize a loss function: measuring the success of a task such as approximation.

Neural Networks (NNs) for numerical PDEs

NNs have been used to solve PDEs in the last three decades. Using DNNs for high-dimensional PDEs emerged in the past few years, and there are many more in developments.

- Use NNs to improve the standard methods: Lee-Kang '90, Yentis-Zaghloul '96, Rudd-Ferrari '15, Tompson-Schlachter-Sprechmann-Perlin '17, Suzuki '17, ...
- Use NNs to approximate the solutions directly, and they may be friendly for high-dimensional problems, such as the physics-informed NN (PINN), Ritz Net, backward-forward SDEs: Dissanayake-Phan-Thien '94, Lagaris-Likas-Fotiadis '98, Beck-E-Jentzen '17, Fujii-Takahashi-Takahashi '17, E-Han-Jentzen '17, He-Li-Xu-Zheng '18, Berg-Nystrom '18, Magill-Qureshi-de Haan '18, Cai-Xu '19, Raissi-Perdikaris-Karniadakis '19, ...
- Use NNs with the variational forms of PDEs, and solve PDEs (SPDEs) by optimization: Nabian-Meidani '18, E-Yu '18, Khoo-Lu-Ying '19, Anitescu-Atroshchenko-Alajlan-Rabczuk '19, Yang-Perdikaris '19, ...

WAN formulation

The weak form of the solution, multiple the equation by a test function φ and perform integration by part,

$$\begin{aligned} &\langle \mathcal{A}[u], \varphi \rangle = 0 \\ &\mathcal{B}[u] = 0, \quad \text{on } \partial\Omega \end{aligned}$$

example: for the elliptic equation,

$$\langle \mathcal{A}[u], \varphi \rangle \triangleq \int_\Omega \sum_{i,j=1}^{d} a_{ij} \partial_j u \partial_i \varphi - f \varphi \, \mathrm{d}x$$

Why weak solution?

(a) Classical solution may not exist.

(b) Integral form is friendly to sample-based computation, which is crucial for high dimension problems.

(c) Solution and test function are in a 2-player game, helping to overcome the challenge of lack of data in neural network training.

A minimax problem

Theorem

Suppose u^* satisfies the boundary condition $\mathcal{B}[u^*] = 0$, then u^* is a weak solution if and only if u^* solves the problem

$$\min_{u \in H^1} \max_{\varphi \in H_0^1} |\langle \mathcal{A}[u], \varphi \rangle|^2 / \|\varphi\|_{H^1}^2.$$

Furthermore, u^* satisfies

$$\|\mathcal{A}[u^*]\|_{op} = 0,$$

where

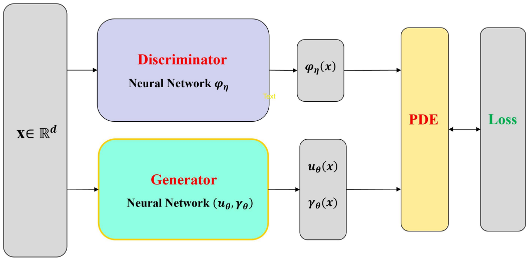
$$\|\mathcal{A}[u]\|_{op} \triangleq \max\{\langle \mathcal{A}[u], \varphi \rangle / \|\varphi\|_{H^1} \mid \varphi \in H_0^1, \varphi \neq 0\},$$

WAN framework

Idea:

- Weak solution $u \in H^1$, approximated by the primary NN u_θ,
- Test function $\varphi \in H_0^1$, approximated by the adversarial NN φ_η.
- Iteratively learn θ to minimize $\|\mathcal{A}[u_\theta]\|_{op}$ with fixed φ_η, and challenges u_θ by maximizing $\langle \mathcal{A}[u_\theta], \varphi_\eta \rangle$ modulus its own norm $\|\varphi_\eta\|_{H^1}$ for every given u_θ .

Weak Adversarial Network



Loss Functions

The lost function used for training (optimization for the parameters) may have many different choices. For example, the following one is used in our computations,

$$\min_{\theta} \max_{\eta} L(\theta, \eta), \quad \text{where} \quad L(\theta, \eta) \triangleq L_{\text{int}}(\theta, \eta) + \alpha L_{\text{bdry}}(\theta),$$

with

$$L_{\text{int}}(\theta, \eta) \triangleq \log |\langle \mathcal{A}[u_\theta], \varphi_\eta \rangle|^2 - \log \|\varphi_\eta\|^2_{H^1},$$

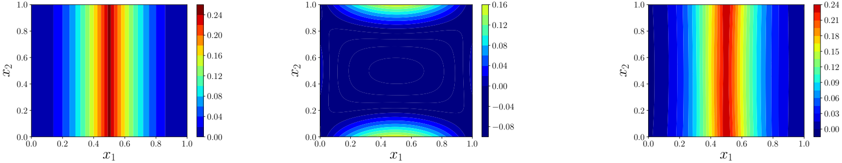
and

$$L_{\text{bdry}}(\theta) \triangleq (1/N_b) \cdot \sum_{j=1}^{N_b} |u_\theta(x_b^{(j)}) - g(x_b^{(j)})|^2.$$

Weak solution v.s. classical solution

$$\begin{cases} \Delta u = 2, & \text{in } \Omega \\ u = g, & \text{on } \partial\Omega \end{cases}$$

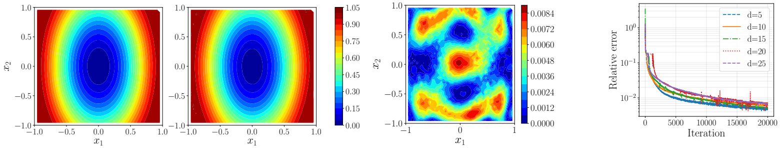
Weak solution exists, but the classical solution doesn't.



(a) True u^* (b) u_s from classical form (c) u_w from weak form

Nonlinear equation ($d = 20$)

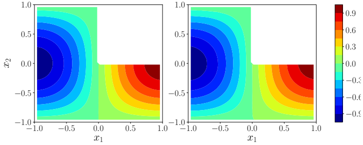
$$\begin{cases} -\nabla \cdot (a(x)\nabla u) + \frac{1}{2}|\nabla u|^2 = f(x) & \text{in} \quad \Omega \triangleq (-1,1)^d, \\ u(x) = g(x) & \text{on} \quad \partial\Omega \end{cases}$$



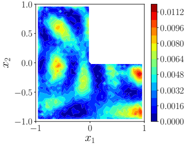
(a) u^* vs u_θ (b) $|u_\theta - u^*|$ (c) Error vs iteration

L-shape domain ($d = 10$)

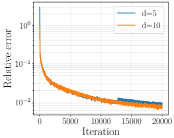
$$
\begin{cases}
-\nabla \cdot (a(x)\nabla u) = f(x) & \text{in} \quad \Omega \triangleq (-1,1)^d \setminus [0,1)^d \\
u(x) = g(x) & \text{on} \quad \partial\Omega
\end{cases}
$$



(a) u^* vs u_θ



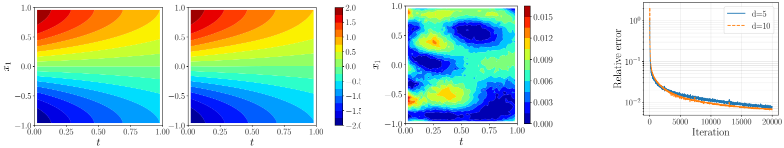
(b) $|u_\theta - u^*|$



(c) Error vs iteration

Time dependent equation ($d = 5$)

$$\begin{cases} u_t - \Delta u - u^2 = f(x,t), & \text{in } \Omega \times [0,T] \\ u(x,t) = g(x,t), & \text{on } \partial\Omega \times [0,T] \\ u(x,0) = h(x), & \text{in } \Omega \end{cases}$$



(a) u^* vs u_θ (b) $|u_\theta - u^*|$ (c) Error vs iteration

Features

- The primary and adversarial NNs are used to train each other. No training data is needed.
- It is flexible in sampling points used to compute the integrals. It fits the frameworks of un-supervised or supervised learning.
- It is mesh-less, basis-less.
- It seeks convergence only in u_θ.
- It is different from existing methods (FEM, Spectral, FDM, Collocation), not Galerkin based, no triangulation, no finite element basis or Fourier basis, no enforcement on selected points.

Inverse problem

Goal: numerically solve inverse problems in high dimensions.

The PDEs in a high-dimension space $\Omega \in \mathbb{R}^d$:

$$\begin{cases} \mathcal{A}[u,\gamma] = 0, & \text{in } \Omega \\ \mathcal{B}[u,\nabla u,\gamma] = 0, & \text{on } \partial\Omega \end{cases} \tag{1}$$

where $\mathcal{A}[u,\gamma]$ may be a second order elliptic differential operator, such as the electrical impedance tomography (EIT) $\mathcal{A}[u,\gamma] = -\nabla\cdot(\gamma\nabla u) - f$, $\mathcal{B}[u,\nabla u,\gamma]$ is the boundary value, u the solution and γ the coefficient function.

The inverse problem: Given the observations of $\mathcal{B}[u,\nabla u,\gamma]$ on $\partial\Omega$, find (u,γ) that satisfies the equation (1).

Challenges: ill-posedness; instability; curse-of-dimensionality. (Alessandrini 1987, Mandache 2001).

Recent deep learning approaches for inverse problems

DNNs have been used for solving inverse problem in the last three decades. Here are partially selected works:

Martin-Choi '15, Tan-Lv-Dong-Takei '18, Yao-Wei-Jiang '19.
Martin-Choi '17, Kang-Min-Ye '17, Jin-Mccann-Froustey-Unser '17, Hamilton-Hauptmann '18, Antholzer-Haltmeier-Schwab '19, Wei-Liu-Chen '19.
Adler-öktem '17, Li-Schwab-Antholzer-Haltmeier '20.
Dadvand-Lopez-Onate '06, Khoo-Ying '18,
Raissi-Perdikaris-Karniadakis '19, Fan-Ying '19,
Jo-Son-Hwang-Kim '19, Bar-Sochen '19, and many more.

The equivalent minimax problem for the inverse problem

Define an operator norm

$$\|\mathcal{A}[u,\gamma]\|_{op} \triangleq \max\{\langle \mathcal{A}[u,\gamma],\varphi\rangle / \|\varphi\|_{H^1} \mid \varphi \in H_0^1, \varphi \neq 0\},$$

Theorem

Suppose (u^*, γ^*) satisfies the boundary condition $\mathcal{B}[u^*, \nabla u^*, \gamma^*] = 0$, then u^* is a weak solution if and only if (u^*,γ^*) solves the problem

$$\min_{u\in H^1,\gamma\in L^2} \max_{\varphi\in H_0^1} |\langle \mathcal{A}[u,\gamma],\varphi\rangle|^2 / \|\varphi\|_{H^1}^2.$$

Furthermore, (u^*, γ^*) satisfies

$$\|\mathcal{A}[u^*,\gamma^*]\|_{op} = 0.$$

WAN framework for inverse problems

Idea:

- Weak solution $u \in H^1$, $\gamma \in L^2$ approximated by the primary NN u_θ and γ_θ respectively.
- Test function $\varphi \in H_0^1$, approximated by the adversarial NN φ_η.
- Iteratively learn θ to minimize $\|\mathcal{A}[u_\theta, \gamma_\theta]\|_{op}$ with fixed φ_η, and challenges u_θ and γ_θ by adjusting φ_η to maximize $\langle \mathcal{A}[u_\theta, \gamma_\theta], \varphi_\eta \rangle / \|\varphi_\eta\|_{H^1}$ for every given $(u_\theta,\, \gamma_\theta)$.

The framework for the inverse problems is almost identical to that for the forward problem.

Theorem

For any $\varepsilon > 0$*, let* $\{\theta_j\}$ *be a sequence of the network parameters in* $(u_\theta, \gamma_\theta)$ *generated by the stochastic gradient descent (SGD) algorithm with integrals in* $\nabla_\theta L(\theta)$ *approximated by sample averages with sample complexities* $N_r, N_b = O(\varepsilon^{-1})$ *in each iteration, then* $\min_{1 \le j \le J} \mathbb{E}[|\nabla_\theta L(\theta_j)|^2] \le \varepsilon$ *after* $J = O(\varepsilon^{-1})$ *iterations.*

- This is the so-called ε-convergence.
- It ensures an approximation to a stationary point only.

Key implementation issues

- Various optimization methods can be used for gradient descent or ascent. We use AdaGrad for the test NN and Adam for solution NN. Auto-differentiation is used to calculate derivatives.

- Use fully-connected feed-forward NNs for both the solution u_θ and the test function φ_η. u_θ has 6 hidden layers with 40 neurons per hidden layer, while φ_η consists of 8 hidden layers with 40 neurons per hidden layer. (Other NN structures can be used as well.)

- Calculate integrals by Monte Carlo method.

- Enforce $\varphi_\eta = 0$ on the boundary by setting $\varphi_\eta = wv_\eta$, where $w = 0$ is pre-selected taking zero on $\partial\Omega$, v_η can be non-zero on the boundary.

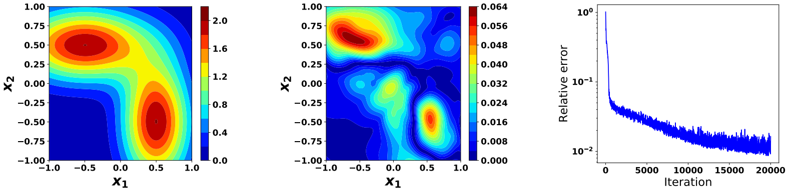
- Other loss functions may work too.

EIT with smooth conductivity (d=5, noise free)

$$-\nabla \cdot (\gamma \nabla u) - f = 0, \quad \text{in } \Omega = (-1,1)^d \qquad (2)$$

$$u - u_b = 0, \; \gamma - \gamma_b = 0, \; \partial_{\vec{n}} u - u_n = 0, \quad \text{on } \partial\Omega \qquad (3)$$

where the conductivity γ is a smooth function. ($N_r = 10^5$, $N_b = 100d$.)

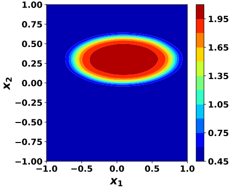


(a) True γ^* (b) $|\gamma^* - \gamma_\theta|$ (c) Error vs iteration

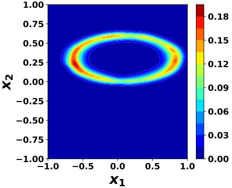
EIT with nearly piecewise conductivity (noise free)

$$-\nabla \cdot (\gamma \nabla u) - f = 0, \quad \text{in } \Omega = (-1,1)^d \tag{4}$$
$$u - u_b = 0,\ \gamma - \gamma_b = 0,\ \partial_{\vec{n}} u - u_n = 0, \quad \text{on } \partial\Omega \tag{5}$$

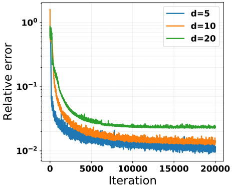
where the conductivity γ is a nearly piecewise function. (For different dimension d, $N_r = 20000d$, $N_b = 100d$.)



(a) True γ^*



(b) $|\gamma^* - \gamma_\theta|$ for $d = 10$



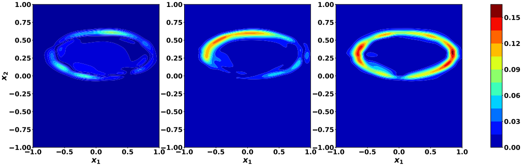
(c) Error vs iteration

EIT with nearly piecewise conductivity (with noise)

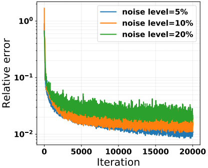
The problem is the same as that defined in (4) and (5),

$$-\nabla \cdot (\gamma \nabla u) - f = 0, \quad \text{in } \Omega = (-1,1)^d$$
$$u - u_b = 0,\ \gamma - \gamma_b = 0,\ \partial_{\vec{n}} u - u_n = 0, \quad \text{on } \partial\Omega$$

where $d = 5$. ($N_r = 20000d$, $N_b = 100d$.)



(a) $|\gamma^* - \gamma_\theta|$ for noise level$= 5\%, 10\%, 20\%$



(b) Error vs iteration

EIT with nonconvex conductivity (d=5, noise free)

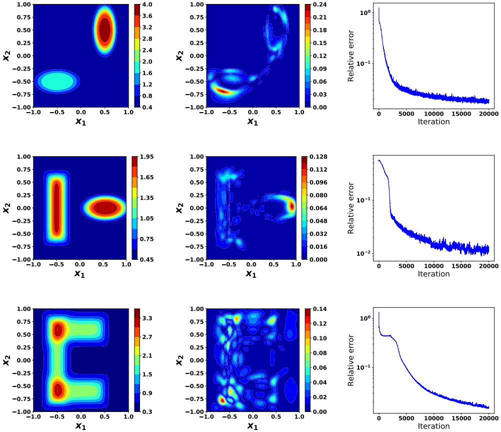
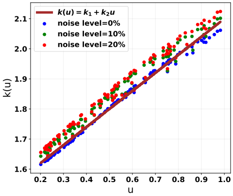


Figure: Left: True γ^*; Middle: $|\gamma^* - \gamma_\theta|$; Right: Error vs iteration

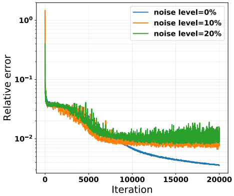
Inverse thermal conductivity problem (d=5, with noise)

$$\partial_t u - \nabla\cdot(\gamma\nabla u) - f = 0, \quad \text{in } \Omega_T = \Omega\times[0,1]$$
$$u - u_i = 0, \quad \text{in } \Omega\times\{0\}$$
$$\nabla u\cdot\vec{n} - u_n = 0,\ u - u_b = 0,\ \gamma - \gamma_b = 0, \quad \text{on } \partial\Omega\times[0,1]$$

where $\gamma(u) = k_1 + k_2 u$ with $k_1 = 1.5$ and $k_2 = 0.6$. ($N_r = 10^5$, $N_b = 100d$.)



(a) $\gamma_\theta(u_\theta)$ vs u_θ



(b) Error vs iteration

Figure: inverse thermal conductivity problem with noise level= 0%, 10%, 20%.

Conclusion and Questions

- A minimax framework for PDEs.
- Using NN in high dimensions.
- A lot of open questions
  - Convergent? Experiments indicate so.
  - Accuracy? Examples are promising.
  - Stability? Seems to be stable, no regularizer is used!
  - Speed? There are rooms to improve.
- Improvement strategies are desirable.

References

- Bao G, Ye X, Zang Y, Zhou H. Numerical Solution of Inverse Problems by Weak Adversarial Networks. Inverse Problems, Vol 36, No. 11, 2020.
- Zang Y, Bao G, Ye X, Zhou H. Weak adversarial networks for high-dimensional partial differential equations. Journal of Computational Physics, Vol 411, 15 June 2020, 109409

Thank you