"New Electricity" for Partial Differential Equations: A Deep Learning Approach

Kailai Xu, Bella Shi, Shuyi Yin
{kailaix, bshi, syin3}@stanford.edu

Overview

Novel Deep Learning Approach for the Laplace equation

$$\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), \boldsymbol{x} \in \Omega \subset \boldsymbol{R}^d$$

$$u(\boldsymbol{x}) = g_D(\boldsymbol{x}), \boldsymbol{x} \in \partial\Omega$$

Motivation

- Traditional methods in engineering are good at $d = 1, 2, 3$;
- For high dimensions, they suffer from **the curse of the dimensionality;**
- Deep Learning is a very promising approach.

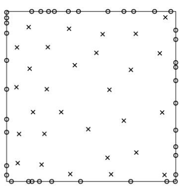
Datasets

- **Generate Boundary Data (** o **)**

$\{(x_i, y_i) \in \partial\Omega\}$, $(g_D)_i = g_D(x_i, y_i)$

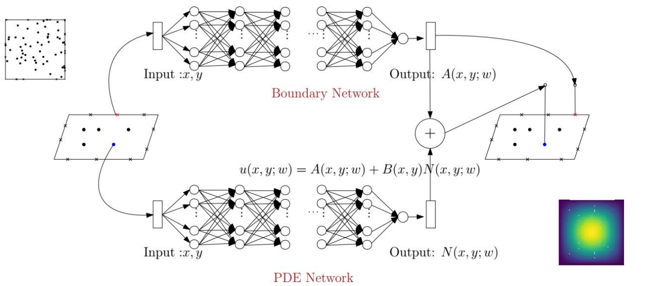
- **Generate Domain Data (** x **)**

$\{(x_i, y_i) \in \Omega\}$, $f_i = f(x_i, y_i)$



Data Generation: A 2D Example.

Models



Boundary Network $A(x, y; w)$

Approximation **on the boundary**

PDE Network $N(x, y; w)$

Coupled with $A(x, y; w)$

Approximation **within the domain**

Loss Function

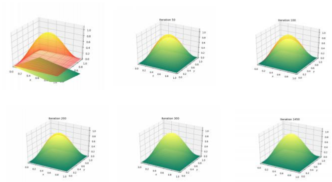
$$\sum_{i=1}^{m}\left((g_D)_i - u(x_i, y_i)\right)^2 + \sum_{i=1}^{n}\left(f_i - \Delta u(x_i, y_i)\right)^2$$

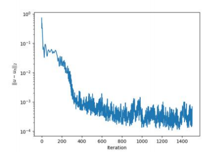
Training Algorithm (GAN style):

```
for number of training iterations
    for k steps do
        sample minibatch on the boundary
        train the boundary network
    end for
    sample minibatch within the domain
    train the PDE network
end for
```

Results

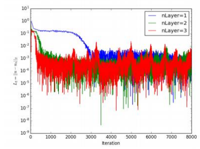
- **PDE:** $\Delta u(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$
- **Boundary condition:** $u(x, y) = g_D(x, y) = 0, x, y \in \partial[0,1]^2$
- **Exact Solution:** $u(x, y) = sin(\pi x)sin(\pi y), f(x, y) = -4\pi^2 u(x, y)$



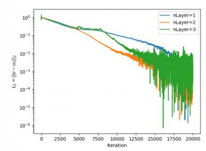


Mean Square Error

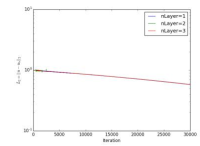
Discussion



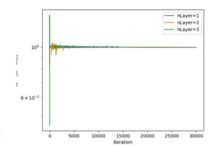
dimension=2, L = 1,2,3



dimension=5, L = 1,2,3



dimension=7, L = 1,2,3



dimension=10, L = 1,2,3

Insights

- For small dimensions, increasing #layers does not increase accuracy, but accelerate convergence.
- For large dimensions, more iterations in training are needed to see convergence, while increasing #layers may also accelerate convergence.

Future Work

- Generalize results to other types of PDEs.
- Investigate algorithms for ill-behaved solutions, such as peaks, exploding gradients, oscillations, etc.

References

[1] I.E. Lagaris, A. Likas and D.I. Fotiadis. *Artificial Neural Networks for Solving Ordinary and Partial Differential Equations*, 1997.

[2] Justin Sirignano and Konstantinos Spiliopoulos. *DGM: A deep learning algorithm for solving partial differential equations*, 2007.