SELF-ADAPTIVE PHYSICS INFORMED NEURAL NETWORKS

Anvita Bhagavathula, Maxim Beekenkamp, Philip LaDuca

BROWN

INTRODUCTION

Physics Informed Neural Networks (PINNs), originally proposed by Raissi, Perdikaris, and Karniadakis, are domain-informed neural models that can approximate solutions to partial differential equations (PDEs) (Raissi et al., 2019). This is achieved by capitalising on the idea that feedforward neural networks are universal function approximators. Specifically, a custom loss function that enforces the structure of the solution to a PDE and its boundary condition behaviour is integrated into the model architecture to do so. The resulting model is a data-efficient function approximator that automatically encodes physical laws into its outputs. For our final project, we first implemented a simple PINN that solves a one-dimensional differential equation. We then added a self-adaptive element to this model that optimally regularizes the boundary components of the loss function during training. Self-adaptive PINNs are able to generate high-accuracy solutions even when provided with a sparse set of input data. This simple but powerful architecture has the potential to solve PDEs in fields ranging from fluid mechanics, biology, and quantum physics and hone our understanding of complex physical phenomena.

METHODOLOGY

PROBLEM SETUP AND DATA

Our goal was to implement a PINN that would produce a solution to a one-dimensional time-independent differential equation with specified boundary conditions. Mathematically, our problem can be represented as follows:

$$\nu u_{xx} - u = e^x \quad \begin{cases} u(-1) = 1 \\ u(1) = 0 \\ x \in [-1, 1] \\ \nu = 10^{-3} \end{cases}$$

Our goal is use our PINN to recover the structure and values of u(x) between for values of x ranging from -1 to 1. The input data for our model is a one-dimensional array of *n* equally spaced x values in the defined range [-1, 1] These are known as collocation points. We implemented this model using Jax.

SIMPLE PINN: ARCHITECTURE

Each component of the differential equation (e.g. u_{xx}) is defined as its own feedforward neural network with randomly initialised weights and biases. Each network has an input layer, output layer, and three hidden layers of size 20. We apply a hyperbolic tangent activation to each layer. We pass our inputs through each component network and compute a customised loss that enforces the structure of the solution at each iteration during training. This loss is the sum of the mean-squared error of the function residue loss and the boundary loss:

$$\mathrm{MSE}_{residue} = \frac{1}{n}\sum[u_{xx} - u - e^x]^2$$
$$\mathrm{MSE}_{upperbound} = u(1)^2$$
$$\mathrm{MSE}_{lowerbound} = [u(-1) - 1]^2$$
$$\mathrm{Loss} = \mathrm{MSE}_{residue} + \mathrm{MSE}_{upperbound} + \mathrm{MSE}_{lowerbound}$$

SELF-ADAPTIVE PINN: ARCHITECTURE

We set up the architecture of the self-adaptive PINN in the same way as we did for the simple PINN. The only modification we made was adding regularizing lambdas to our boundary losses that were also trained. We added an L-BFGS-B minimizer at the end of training to further optimizer network parameters

$$\mathrm{Loss} = \mathrm{MSE}_{residue} + \lambda_1 MSE_{upperbound} + \lambda_2 MSE_{lowerbound}$$

RESULTS

We trained both our simple PINN and the self-adaptive PINN for approximately 20,000 iterations, or epochs, using an Adam optimizer with a learning rate of 5e-4. We also used 40 collocation points which meant that our output was also an array of 40 points representing the solution within our specified domain. Below, we display each PINN's predicted solution to our differential equation and its boundary and residue loss over the epochs. We also display the final loss of each model and a visual comparison of each model's solution.

SIMPLE PINN RESULTS

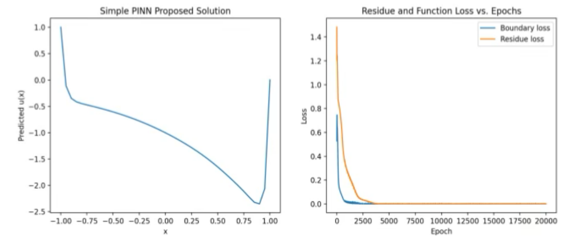

Fig 1: The simple PINN's proposed solution and its loss over the epochs.

The final combined boundary and residue loss on our simple PINN's solution was 1.29e-5.

SELF-ADAPTIVE PINN RESULTS

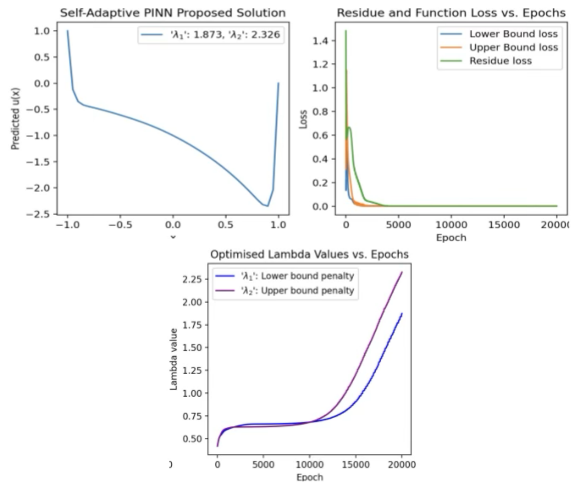

Fig 2: The self-adaptive PINN's proposed solution, its loss, and its lambda values over the epochs.

The final combined boundary and residue loss on our self-adaptive PINN's solution was 2.25e-6. We also observe that the loss decreases as the penalty terms increase in value, indicating that regularizing the loss increases accuracy.

COMPARISON

While visually, the predicted solutions look similar, we see that the self-adaptive PINN's solution has a much lower final loss. This indicates that the adaptive training strategy has the potential to produce higher accuracy solutions overall.

DISCUSSION

We were able to successfully implement a physics-informed neural network to predict the solution to a one-dimensional differential equation. Some advantages of our implementation include the fast training time due to Jax and the high-accuracy of our predicted solutions. However, there are still several limitations to our implementation, that are discussed further below.

- While the self-adaptive PINN was able to reduce the overall loss on the predicted solution, it does not solve a fundamental issue with the PINN framework: the smoothness/accuracy of the solution is limited by the number of collocation points we use.
- Our work only solves a one-dimensional time-independent differential equation. However, most physical phenomena are time dependent and occur in 3D. Producing solutions for these PDEs would require modifying our implementation in some manner. An example of time-dependent solutions generated by a PINN for the Schrodinger equation are provided in Fig 3

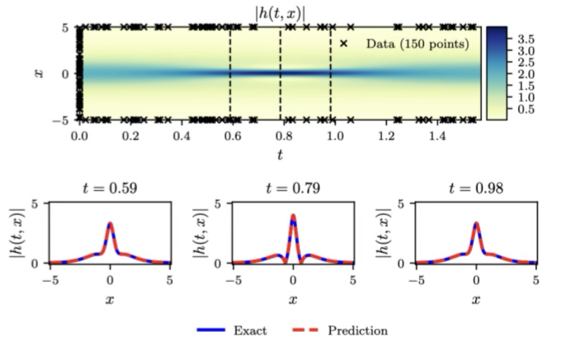

Fig 3: PINN generated solution to Schrodinger equation over time (Raissi, et al., 2019)

FUTURE WORK

The first way we hope to expand our work is by applying self-adaptive learning to 2D differential equations. Additionally, we would also like to explore how we can use self-adaptive learning to optimize the number of collocation points required to generate high-accuracy solutions. This work could have important consequences to applying PINNs to solve data-poor problems in the sciences.

REFERENCES

- Raissi, M., Perdikaris, P., &amp; Karniadakis, G. E. (2019). Physics-informed Neural Networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686–707.
- McClenny, L. (n.d.). Self-adaptive physics-informed neural networks using a soft ... - arxiv. from https://arxiv.org/pdf/2009.04544.pdf