Solving Differential Equations Through Neural Networks.

Nick Baumgartner, Melissa Butler, Paul Sansah
University of Wyoming, Laramie, WY

COSC 5555 - Introduction to Machine Learning, Spring 2022
University of Wyoming
May 13, 2022

Additional support, help, and resources from Silba Dowell

Problem Statement

In the study of differential equations there are many techniques, both analytical and numerical, that are used to find unique solutions. Most of the more complicated equations do not have direct analytic solutions and numerical techniques are the standard to approximate solutions. An emerging field applies deep learning to solving differential equations through Physics Informed Neural Networks (PINNS). These are neural networks that are trained using a set of points in the domain and rely on the physics of the equation to inform the loss function. The goal of this work is to explore this relatively new technique for finding the solution of Ordinary Differential Equations (ODEs), Initial Value Problems (IVPs), and Boundary Value Problems (BVPs). We will look at current literature, prevalent methodology, and showcase the results of PINNs on various ODEs.

Significance

- Most natural phenomena can be modelled through PDEs and ODEs, e.g. fluid dynamics, mechanical vibrations, electrical systems, etc..
- Classic numerical techniques for approximating solutions suffer from high computation time, domain discretization difficulties, and the curse of dimensionality.
- Deep learning via PINNs uses the compositional nature of neural networks and automatic differentiation to overcome many of the classic difficulties.
- The need for more efficient, accurate, more stable approximations is increasing and machine learning is tackling problems from a new and exciting perspective.

Literature Survey

- *DeepXDE: A Deep Learning Library for Solving Differential Equations* by Lu et al. [1]
  - DeepXDE - Python library for research and education
  - Analysis - existence, error, and possible overfitting
  - Residual-Based Adaptive Refinement (RAR) - adaptive training domain
- *Solving high-dimensional partial differential equations using deep learning* by Han et al. [2]
  - Variational Formulation - backward stochastic differential equations
  - Curse of Dimensionality - computation cost increases exponentially with dimensions
  - Non-linearity - Black-Scholes, Hamilton-Jacobi-Bellman, and Allen-Cahn equations
- More work is needed for ODEs

Methodology

Our goal is to approximate the function $y : \Omega \subseteq \mathbb{R} \to \mathbb{R}$ that satisfies,

$$\begin{cases} y'(x) = f(x, y) \\ y(x_0) = y_0 \in \mathbb{R} \end{cases}.$$

Our approximate solution is

$$y(x) \approx \hat{y}(x; P) = \sum_{j=1}^{M} h(xw_0[j] + b_0[j])w_1[j] + b_1,$$

where $w_0, b_0, w_1 \in \mathbb{R}^M$ and $b_1 \in \mathbb{R}$ are our parameters, P, to be trained and $h : \mathbb{R} \to \mathbb{R}$ is our activation function,

$$h(z) = \frac{1}{1 + e^{-z}}.$$

Methodology

Let,

$$X = \{x_i\}_{i=1}^{n} \subset \Omega$$

be our dataset and define our loss function $L : \mathbb{R}^n \times \mathbb{R}^{3M+1} \to \mathbb{R}$, as

$$L(\hat{y}(X;P)) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial}{\partial x_i}\hat{y}(x_i;P) - f(x_i,\hat{y}(x_i;P))\right)^2 + \left(\hat{y}(x_0;P) - y_0\right)^2.$$

So, $\frac{\partial L}{\partial P} : \mathbb{R}^n \to \mathbb{R}^m$ is a vector valued function defined as,

$$\frac{\partial L}{\partial P} = \left(\frac{\partial L}{\partial P_1}, \frac{\partial L}{\partial P_2}, \ldots, \frac{\partial L}{\partial P_{3M+1}}\right).$$

Methodology

We want to find

$$\arg\min_{P} L(\hat{y}(X;P)),$$

and use a gradient descent optimization. Initialize P and several training variables, $m, \eta \in \mathbb{R}$ and $v \in \mathbb{R}^{3M+1}$. For each forward pass we calculate G

$$G = \frac{\partial L}{\partial P}(\hat{y}(X, P + mv)).$$

Then for the backward pass we update,

$$v = mv - \eta G$$
$$P = P + v.$$

Methodology

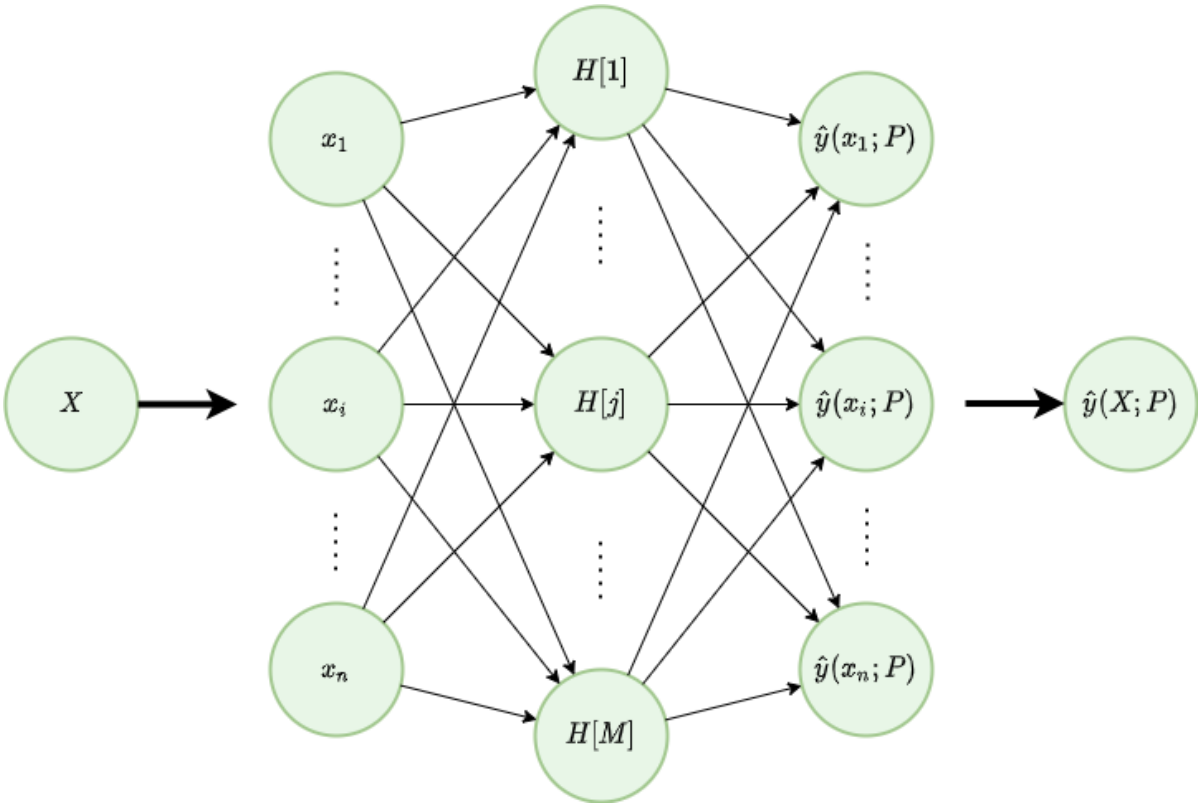

Figure: Single Layer Neural Network for $\hat{y}(X; P)$

$$H[j] = [h(x_1 w_0[j] + b_0[j]), h(x_2 w_0[j] + b_0[j]), \ldots, h(x_n w_0[j] + b_0[j]]^\top,$$

Results

Three distinct differential equations

- First order IVP:

$$\begin{cases} \dot{y} = 2xy \\ y(0) = 1 \end{cases}$$

- Second Order BVP with Analytic Solution:

$$\begin{cases} -u'' - \sin(2x)\cos(u) + 2u = -2 - \sin(2x)\cos(x^2 - 1) + 2(x^2 - 1) \\ u(-1) = 0, \quad u(1) = 0 \end{cases}$$

- Second Order BVP with no closed Analytic Solution:

$$\begin{cases} u'' + \lambda u = 0 \\ u(0) = 0, \quad u(L) = 0 \end{cases}$$

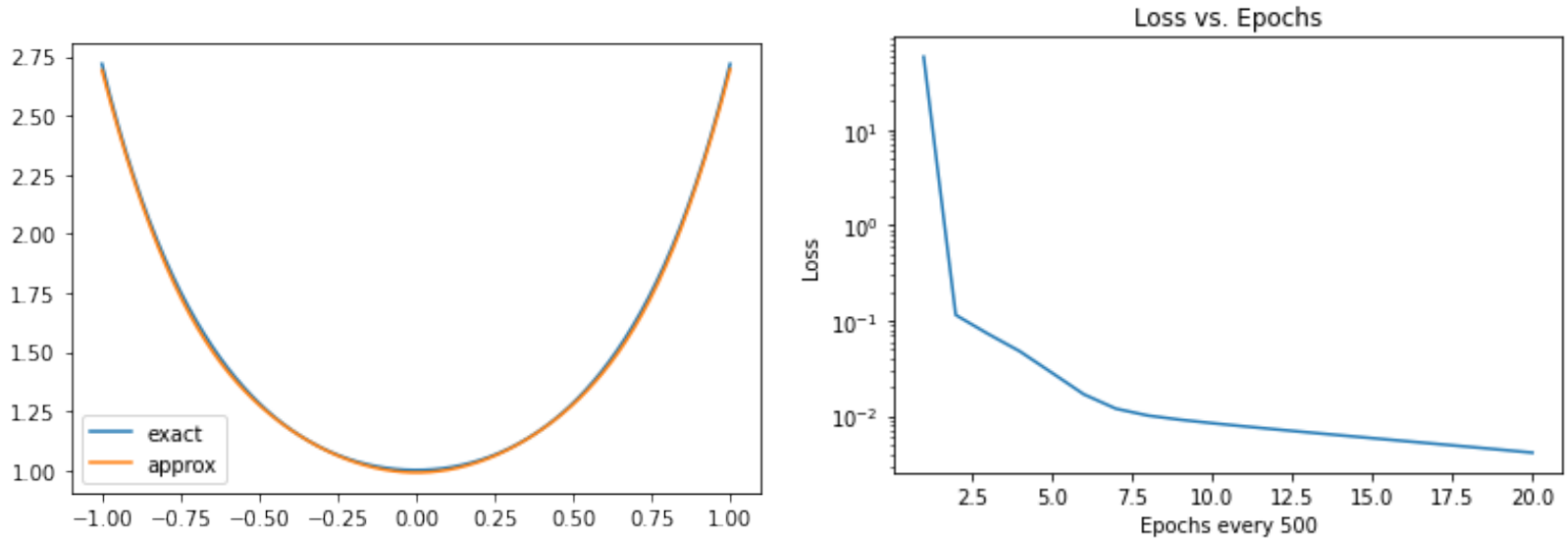

Figure: First Order IVP

The exact solution to the system is $y = \exp(x^2)$ and the approximate solution presented is after 10,000 epochs run.

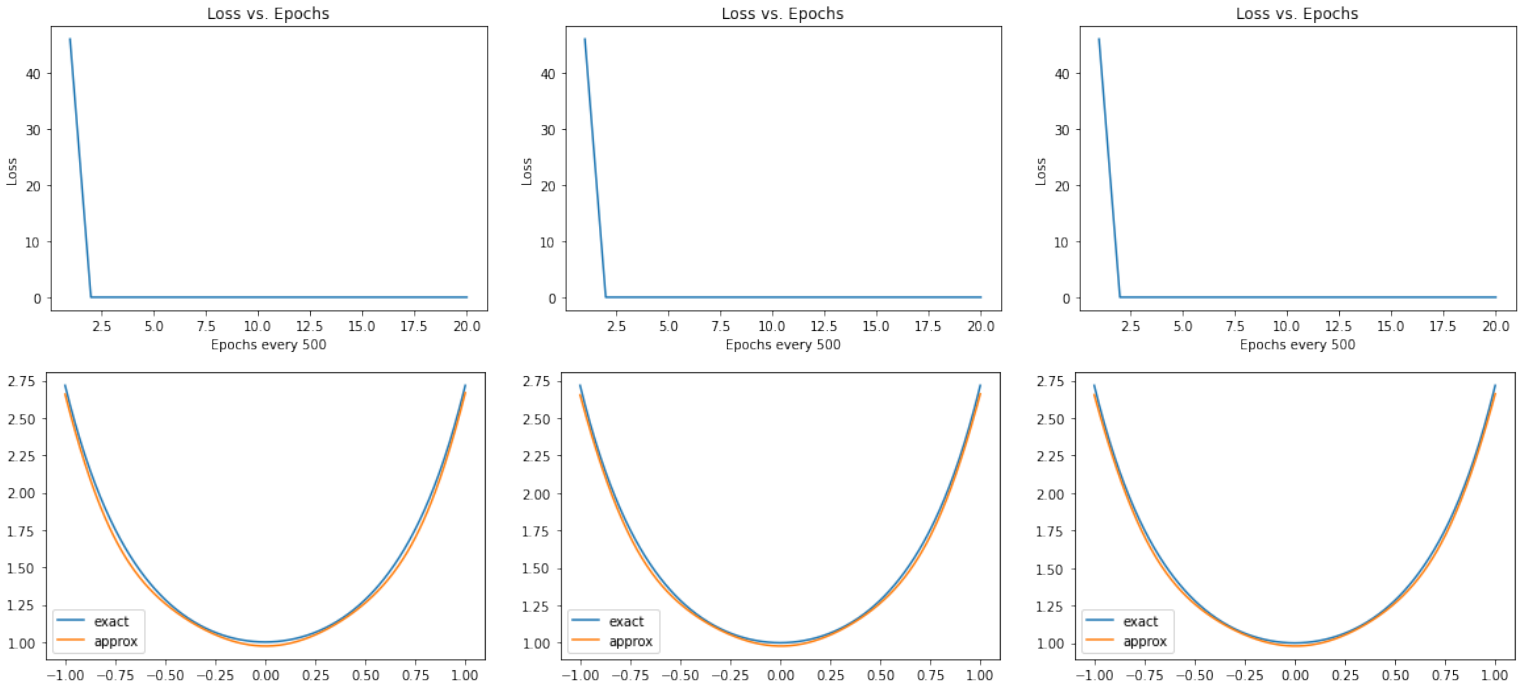

Figure: Loss functions and results for n = 11,101,401 training points

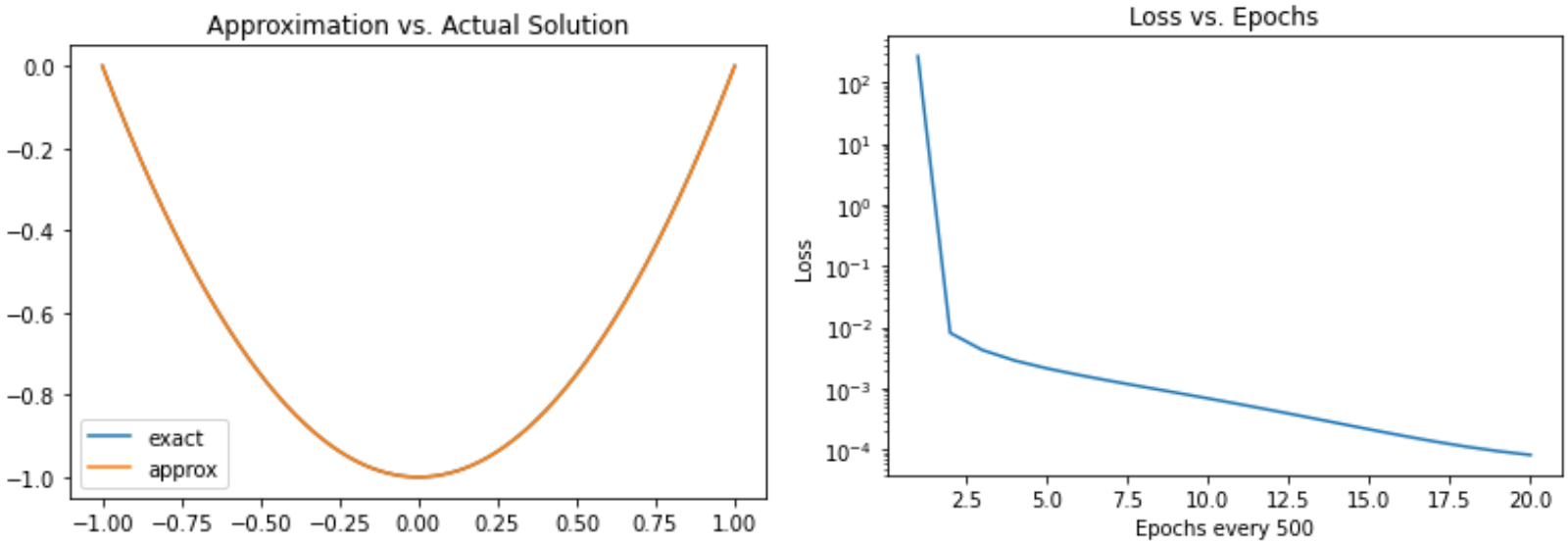

Figure: Second Order BVP

The exact solution to the system is $u = x^2 - 1$ and the approximate solution presented is after 10,000 epochs run.

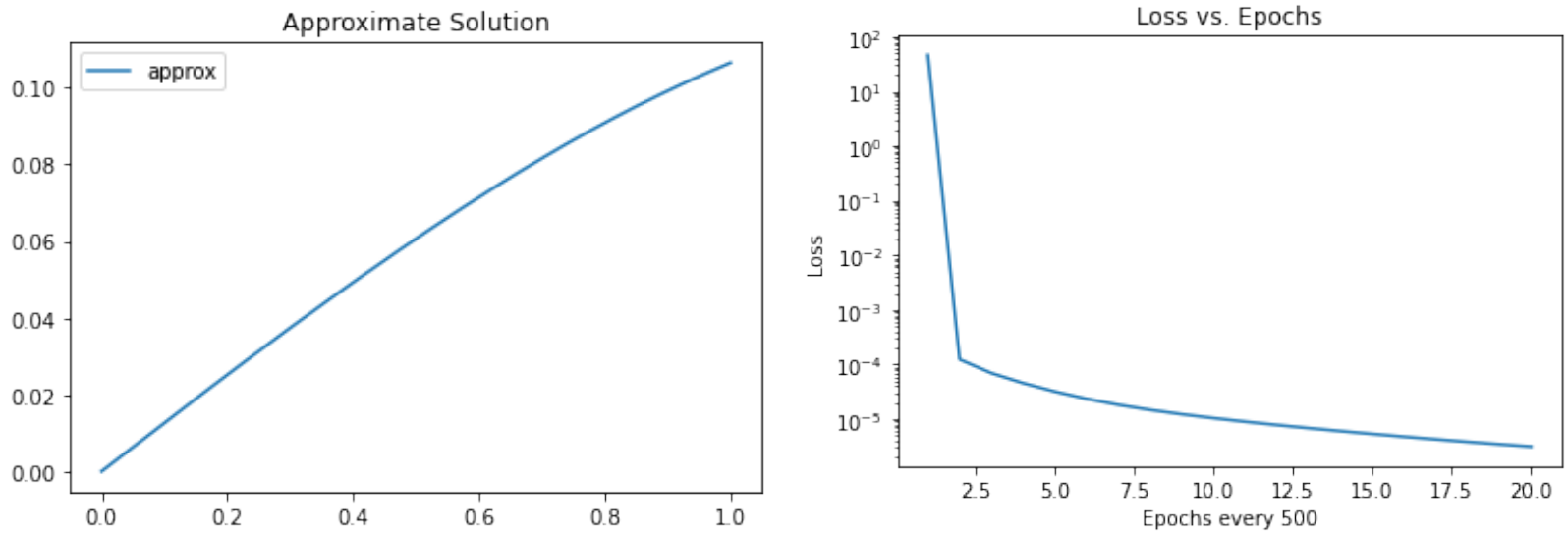

Figure: Second Order BVP

The above plot represents that of the wave equation, typically a PDE, with constant acceleration and constraints on the boundary which reduce it to an ODE whose analytic solution is that of an infinite series.

Challenges

While each aspect of the project presented its own challenges that ranged from coding issues to concept understanding the largest challenges were:

- Package Management
- Boundary Conditions
- Training size manipulation

Conclusions and Future Work

- Overall Accuracy
- Number of Nodes
- Expansion into PDE
- Exploration of IVP with derivative condition

References

Lu, Lu and Meng, Xuhui and Mao, Zhiping and Karniadakis, George Em, DeepXDE: A deep learning library for solving differential equations, SIAM Review 63:: 208-228.

Han, Jiequn and Jentzen, Arnulf and E, Weinan, Solving high-dimensional partial differential equations using deep learning, National Academy of Sciences 115: 8508-8510.

Tobias Knoke and Thomas Wick, Solving differential equations via artificial neural networks: Findings and failures in a model problem, https://www.sciencedirect.com/science/article/pii/S2666657X21000197.

Tamirat Temesgen Dufera, Deep Neural Network, Algorithm, Systems of ordinary differential equation, https://www.sciencedirect.com/science/article/pii/S2666827021000293.

Gillgren, Simon, Solving Differential Equations with Neural Networks, https://github.com/Gillgren/Solving-Differential-Equations-with-Neural-Networks

THANK YOU