USING NEURAL NETWORKS TO SOLVE PARTIAL DIFFERENTIAL EQUATION

JOSHUA GEORGE

Abstract. This abstract concerns the introduction of some techniques used to solve partial differential equations (PDEs) using neural networks. We provide some background and later talk about the project goals.

1. Introduction

The topic of using neural networks to PDEs has gained significant attention in recent years due to its potential to provide accurate solutions for complex PDEs. We use the approach, known as physics-informed machine learning (PIML), which involves incorporating the governing PDE into the neural network architecture as a constraint. This allows for the network to learn the underlying physics of the system and provide accurate solutions.

2. Background

2.1. **PDEs.** A partial differential equation (PDE) is an equation of the form:

$$F(x_1, x_2, ..., x_n, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, ..., \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, ..., \frac{\partial^2 u}{\partial x_n^2}, ...) = 0$$

where u is the unknown function of n independent variables $x_1, x_2, ..., x_n$, and F is a function of u and its partial derivatives with respect to the independent variables. PDEs are used to model a wide range of physical, biological, and social phenomena, from fluid dynamics to population dynamics to finance. One such example is the heat equation which is a PDE that describes the distribution of heat in a medium over time. It is mathematically given by

$$\partial_t u = \kappa \Delta u$$

where $k > 0$ is a constant, $u = u(t, x), x \in \mathbb{R}^3$ is the temperature function at time t and position x on a physical body, and Δ is called the Laplacian operator. Here Δu is

$$\Delta u = \partial_{x_1 x_1} u + \partial_{x_2 x_2} u + \partial_{x_3 x_3} u.$$

2.2. **Neural Networks.** A neural network is essentially a transformation that maps input data to output data. The function consists of multiple layers of neurons, which represent the basic computational unit that processes and transforms input information. The output of each neuron is determined by a weighted sum of the inputs, followed by the application of an activation function. The weights and biases of the neurons are learned through a process called backpropagation, which involves minimizing a loss function that measures the error between the predicted output and the actual output. The so-called activation functions are used to introduce non-linearity into the network, allowing it to model complex relationships between the input and output. Mathematically, the output from the neural network is

$$f_1 \left(f_2 \left(\dots f_{H-1} \left(f_H \left(\mathbf{x} \mathbf{W}^{(H)} \right) \mathbf{W}^{(H-1)} \right) \dots \right) \mathbf{W}^{(1)} \right)$$

where $\mathbf{x}$ is the input denote each activation function $f_1, \dots, f_H$, ordered with f_1 as the output activation, and $p_1, \dots, p_{H-1}$ as the hidden dimensions with $H-1$ hidden layers and $\mathbf{W}^{(1)} \in \mathbb{R}^{p_1 \times m}, \mathbf{W}^{(2)} \in \mathbb{R}^{p_2 \times p_1}, \dots, \mathbf{W}^{(H)} \in \mathbb{R}^{d \times p_{H-1}}$ the wright matrices. Then

3. Project Goals

My main goal is to study the following methods for solving a specific type of PDE and compare their results. This could be done by either generating synthetic data or using existing data available online to study PDEs of similar forms.

3.1. **Physics Informed Neural Networks (PINNs).** [1] Class of neural network-based models that can be used to solve PDEs. The idea behind PINNs is to use neural networks to learn the solution to a PDE, while also incorporating any known physical laws or constraints that govern the behavior of the system being modeled. The basic approach in PINNs is to use a neural network to approximate the solution to the PDE, and then incorporate the PDE as a constraint in the training process. This is done by adding a loss function that penalizes any deviation from the PDE, which encourages the neural network to learn a solution that satisfies the PDE.

3.2. **Deep Galerkin Method (DGM).** [2] The DGM is a machine learning-based approach for solving PDEs. Traditional numerical methods, require the discretization of the domain into a grid and the solution of a system of linear equations, the DGM approximates the solution of the PDE directly using a neural network. The key idea behind the DGM is to use a deep neural network to represent the solution of the PDE as a function of the input variables, such as the spatial and temporal coordinates. The neural network is trained to minimize a loss function that measures the difference between the predicted and true solutions of the PDE

4. Applications

Due to the recent advances in this field using Neural Networks to solve PDEs has shown promise in various fields such as:

- Fluid dynamics: Neural networks have been used to solve the Navier-Stokes equations, which describe the motion of fluids. This has been applied to problems such as turbulent flow, flow around obstacles, and fluid-structure interactions.
- Quantum mechanics: Neural networks have been used to solve the Schrödinger equation, which describes the behavior of quantum particles. This has been applied to problems such as molecular dynamics and quantum control.
- Heat transfer: Neural networks have been used to solve the heat equation, which describes the transfer of heat in a material. This has been applied to problems such as heat conduction in composite materials and cooling of electronic devices.

5. Possible Limitations

A few limitations we could incur while doing the project are the possibility of the lack of data (data for the domain and boundary conditions of the domain are required), inaccurate and unstable solutions of the PDEs and lack of computing power required for the project.

References

[1] Raissi, M., Perdikaris, P., & 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. (ICML 2017).

[2] Sirignano, J., Spiliopoulos, K. (2018). *DGM: A deep learning algorithm for solving partial differential equations.* In *International Conference on Machine Learning* (pp. 4465-4474).