Physics Informed Neural Networks

A Priyanka, Sudharshan S

M.Sc. Theoretical Computer Science

Department of Applied Mathematics and Computational Sciences

PSG College of Technology, Coimbatore

April 10, 2019

Contents

CHAPTER 1

Introduction

Physics, being wholly dependent on Mathematics, models various phenomenon of nature. One such model is the mathematical formulation of motion of a viscous liquid known as Navier-Stokes equations. The fascination of the interlink of non-linear dynamics and the famous deep learning model, neural network, persuaded this work. The interconnection between the two diverse concepts was studied and published by Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Further, the idea had been elaborated and detailed in Data-driven Discovery of Nonlinear Partial Differential Equations by the authors of earlier papers.

The paper on Data Driven Solutions of Non Linear Partial Differential Equations [1], introduces physics informed neural networks that are trained to solve supervised learning tasks while respecting any given law of physics described by general nonlinear partial differential equations. Depending on the nature and arrangement of the available data, two distinct classes of algorithms, namely continuous time and discrete time models are devised. The resulting neural networks form a new class of data-efficient universal function approximators that naturally encode any underlying physical laws as prior information. In this first part, it demonstrates how these networks can be used to infer solutions to partial differential equations, and obtain physics-informed surrogate models that are fully differentiable with respect to all input coordinates and free parameters

The paper on Data Driven Discovery of Non Linear Equations [2] refines the effectiveness of the approach put forth in Data-Driven Solutions of Nonlinear Partial Differential Equations [1], by demonstrating using a wide range of benchmark problems in mathematical physics, including conservation laws, incompressible fluid flow, and the propagation of nonlinear shallow-water waves.

CHAPTER 2

Literature Review

Before the introduction with PINNs, this chapter helps in familiarizing with few basic models of deep learning, approaches in number theory and a basic of Burger's equation.

2.1 Motivation

Deep learning has gained unprecedented attention over the last few years, and deservedly so, as it has introduced transformative solutions across diverse scientific disciplines. Despite the ongoing success, there exist many scientific applications that have yet failed to benefit from this emerging technology, primarily due to the high cost of data acquisition. It is well-known that the current state-of-the-art machine learning tools are lacking robustness and fail to provide any guarantees of convergence when operating in the small data regime, i.e., the regime where very few training examples are available.

In this study, physics informed neural networks as a viable solution for training deep neural networks with few training examples, for cases where the available data is known to respect a given physical law described by a system of partial differential equations is introduced. Such cases are abundant in the study of physical, biological, and engineering systems, where longstanding developments of mathematical physics have shed tremendous insight on how such systems are structured, interact, and dynamically evolve in time.

Having seen how the knowledge of an underlying physical law can introduce a structure that effectively regularizes the training of neural networks, and enables them to generalize well even when only a few training examples are available. Through the lens of different benchmark problems, the key features of physics informed neural networks is highlighted in the context of data-driven solutions of partial differential equations.

2.2 Burger's Equation

Burgers equation is obtained as a result of combining nonlinear wave motion with linear diffusion and is the simplest model for analyzing the combined effect of nonlinear advection and diffusion. The presence of viscous term helps suppress the wave-breaking, smooth out shock discontinuities, and expect to obtain a well-behaved and smooth solution. Moreover, in the inviscid limit, as the diffusion term becomes vanishingly small, the smooth viscous solutions converge non-uniformly to the appropriate discontinuous shock wave, leading to an alternative mechanism for analyzing conservative nonlinear dynamical processes. The advection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial t} = 0$$

propagates an initial waveform or signal at velocity c while maintaining the precise form of the initial waveform. On the other hand, the nonlinear equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial t} = 0$$

propagates signals in such a way that distortion occurs in the waveform profile; that is, the nonlinear advection term αu_{xx} causes either a shocking or rarefaction effect.

It is well known that the heat equation

$$\frac{\partial u}{\partial t} + \alpha \frac{\partial^2 u}{\partial t^2} = 0$$

2.3 Deep Learning Model

A neural network is just a combination of lots of these units. Each one performs a very simple and stereotyped function, but in aggregate they can do some very useful computations. The simplest kind of feed-forward network is a multilayer perceptron (MLP), as shown in Figure 2.3.1. The perceptron was a particular algorithm for binary classification, invented in the 1950s. Most multilayer perceptrons have very little to do with the original perceptron algorithm. Here, the units are arranged into a set of layers, and each layer contains some number of identical units. Every unit in one layer is connected to every unit in the next layer, then the network is fully connected. The first layer is the input layer, and its units take the values of the input features. The last layer is the output layer, and it has one unit for each value the network outputs (i.e. a single unit in the case of regression or binary classifiation, or K units in the case of K-class classification). All the layers in between these are known as hidden layers because ahead of time what these units should compute is not known, and this needs to be discovered during learning. A MLP could be mathematically designed by considering input as x_j and output unit as y. The units in the l th hidden layer will be denoted $h_i^(l)$. The network is fully connected, so each unit receives connections from all the units in the previous layer. This means each unit has its own bias, and theres a weight for every pair of units in two consecutive

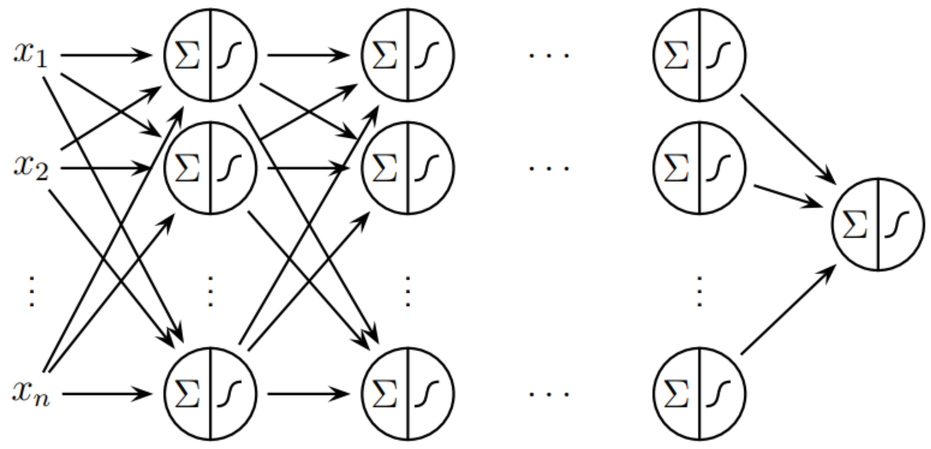

Figure 2.3.1: Multi Layer Neural Network

layers. Therefore, the networks computations can be written out as

$$h_i^{(1)} = \phi^{(1)}\left(\sum_j w_{ij}^{(1)} x_j + b_i^{(1)}\right)$$

$$h_i^{(2)} = \phi^{(2)}\left(\sum_j w_{ij}^{(2)} h_j^{(1)} + b_i^{(2)}\right)$$

$$y_i = \phi^{(3)}\left(\sum_j w_{ij}^{(3)} h_j^(2) + b_i^{(3)}\right)$$

Note: $^{(1)}$ and $^{(2)}$ are different because different layers may have different activation functions.

2.4 Runge-Kutta Method

In the forward Euler method, the information on the slope or the derivative of y is used at the given time step to extrapolate the solution to the next time-step. Local Trucation Error (LTE) for the method is $O(h^2)$, resulting in a first order numerical technique. Runge-Kutta methods are a class of methods which judiciously uses the information on the 'slope' at more than one point to extrapolate the solution to the future time step. Firstly, the derivation of the second order RK method where the LTE is $O(h^3)$. Given the following Initial Value Problem with time step h, and the solution y_n at the n^{th} time step, the solution at y_{n+1} is

$$k_1 = h\ f(y_n, t_n)$$

$$k_2 = h\ f(y_n + \beta k_1, t_n + \alpha h)$$

$$y_{n+1} = y_n + a\ k_1 + b\ k_2$$

where the constants α, β, a and b have to be evaluated so that the resulting method has a LTE $O(h^3)$.

CHAPTER 3

Physics Informed Neural Networks

This part deals on a different approach by employing deep neural networks and leverage their well-known capability as universal function approximators, which directly tackles nonlinear problems without the need for committing to any prior assumptions, linearization, or local time-stepping. The approach is achieved by recent development in automatic differentiation one of the most useful but perhaps underused techniques in scientific computing to differentiate neural networks with respect to their input coordinates and model parameters to obtain physics informed neural networks This chapter deals with predictive inference, filtering and smoothing, or data-driven solutions of partial differential equations.

3.1 Model Overview

Consider parametrized and nonlinear partial differential equations of the general form

$$u_t + \mathcal{N}[u; \lambda] = 0, \tag{3.1}$$

where $u(t, x)$ denotes latent (hidden) solution and $\mathcal{N}[\cdot, \lambda]$ is a nonlinear operator parametrized by λ. This setup encapsulates a wide range of problems in mathematical physics including conservation laws, diffusion processes, advection-diffusion-reaction systems, and kinetic equations. As a motivating example, the one dimensional Burgers' [1] equation corresponds to the case where,

$$\mathcal{N}[u; \lambda] = \lambda_1 u u_x + \lambda_2 u_{xx},$$

where $\lambda = (\lambda_1, \lambda_2)$.Here, the subscripts denote partial differentiation in either time or space. In this chapter, the spotlight is on computing datadriven solutions to partial differential equations of the general form

$$u_t + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T],$$

where $u(t, x)$ denotes the latent (hidden) solution, $\mathcal{N}[.]$ is a nonlinear differential operator, and $\Omega \subseteq \mathbb{R}^D$. Now based on the characteristics of time, the model is subdivided into two different algorithms, namely continuous and discrete time models, and will highlight their properties and performance through the lens of different benchmark problems.

3.2 Continuous Time Model

Define $f(t, x)$ to be given by the left-hand-side of equation

$$f = u_t + \mathcal{N}[u],$$

and proceed by approximating $u(t, x)$ by a deep neural network. This network can be derived by applying the chain rule for differentiating compositions of functions using automatic differentiation.

As an example, consider the Burgers' equation. This equation arises in various areas of applied mathematics, including fluid mechanics, nonlinear acoustics, gas dynamics, and traffic flow. It is a fundamental partial differential equation and can be derived from the Navier-Stokes equations for the velocity field by dropping the pressure gradient term. For small values of the viscosity parameters, Burgers' equation can lead to shock formation that is notoriously hard to resolve by classical numerical methods. In one space dimension, the Burger's equation along with Dirichlet boundary conditions [1] reads as

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0,\ x \in [-1, 1],\ t \in [0, 1],$$
$$u(0, x) = -sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

Define $f(t, x)$ as

$$f = u_t + uu_x - (0.01/\pi)u_{xx},$$

and proceed by approximating $u(t, x)$ by a deep neural network. The shared parameters between the neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss

$$MSE = MSE_u + MSE_f$$

where

$$MSE_u = \tfrac{1}{N_u} \textstyle\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2$$

$$MSE_f = \tfrac{1}{N_f} \textstyle\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Here, $\{t_u^i, x_u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t, x)$ and $\left\{t_f^i, x_f^i\right\}_{i=1}^{N_f}$ specify the collocations points for $f(t, x)$. The loss MSE_u corresponds to the initial and boundary data while MSE_f enforces the structure imposed by equation at a finite set of collocation points. For larger data-sets a more computationally efficient mini-batch setting can be readily employed using stochastic gradient descent and its modern variants. Despite the fact that there is no theoretical guarantee that this procedure converges to a global minimum, our empirical evidence indicates that, if the given partial differential equation is well-posed and its solution is unique, our method is capable of achieving good prediction accuracy given a sufficiently expressive neural network architecture and a sufficient number of collocation points N_f . This general observation deeply relates to the resulting optimization landscape induced by the mean square error loss, defines an open question for research that is in sync with recent theoretical developments in deep learning. The robustness of the method and implementation is given in Chapter xx with the analysis of the results obtained.

3.3 Discrete Time Model

Recall the Runge-Kutta methods with q stages to equation (3.1) and obtain

$$u^{n+c_i} = u^n - \Delta t \textstyle\sum_{j=1}^{q} a_{ij}\mathcal{N}[u^{n+c_j}], i = 1, \ldots, q,$$

$$u^{n+1} = u^n - \Delta t \textstyle\sum_{j=1}^{q} b_j\mathcal{N}[u^{n+c_j}]$$

This general form encapsulates both implicit and explicit time-stepping schemes, depending on the choice of the parameters $\{a_{ij}; b_j; c_j\}$. Proceeding by placing a multi-output neural network prior on

$$[u^{n+c_1}(x), \ldots, u^{n+c_q}(x), u^{n+1}(x)]$$

The physics informed neural network takes the following as its input

$$[u_1^n(x), \ldots, u_q^n(x), u_{q+1}^n(x)].$$

To highlight the key features of the discrete time representation recall the problem of data-driven solution of the Burgers' equation. For this case, the nonlinear operator,

$$\mathcal{N}[u^{n+c_j}] = u^{n+c_j} u_x^{n+c_j} (0.01/\pi) u_{xx}^{n+c_j},$$

and the shared parameters of the neural networks and can be learned by minimizing the sum of squared errors.

$$SSE = SSE_n + SSE_b,$$

where

$$SSE_n = \sum_{j=1}^{q+1} \sum_{i=1}^{N_n} |u_j^n(x^{n,i}) - u^{n,i}|^2,$$

and

$$SSE_b = \sum_{j=1}^{q} (|u^{n+c_i}(-1)|^2 + |u^{n+c_i}(1)|^2 + |u^{n+1}(-1)|^2 + |u^{n+1}(1)|^2).$$

Here, $\{x^{n,i}, u^{n,i}\}_{i=1}^{N_n}$ corresponds to the data at time t_n. The Runge-Kutta scheme now allows one to infer the latent solution $u(t, x)$ in a sequential fashion. Starting from initial data $\{x^{n,i}, u^{n,i}\}_{i=1}^{N_n}$ at time t_n and data at the domain boundaries $x = -1$ and $x = 1$, the aforementioned loss function can be used to train the networks and predict the solution at time t_{n+1}. A Runge-Kutta time-stepping scheme would then use this prediction as initial data for the next step and proceed to train again and predict $u(t^{n+2}, x)$, $u(t^{n+3}, x)$, etc., one step at a time. In classical numerical analysis, these steps are usually confined to be small due to stability constraints for explicit schemes or computational complexity constrains for implicit formulations. These constraints become more severe as the total number of Runge-Kutta stages q is increased, and, for most problems of practical interest, one needs to take thousands to millions of such steps until the solution is resolved up to a desired final time. In sharp contrast to classical methods, here one can employ implicit Runge-Kutta schemes with an arbitrarily large number of stages at effectively no extra cost. This enables one to take very large time steps while retaining stability and high predictive accuracy, therefore allows to resolve the entire spatio-temporal solution in a single step.

The details on how the model is built is specified in the forthcoming section.

3.4 Implementation

Practical experimentation of the papers [1] [2] were conducted,for Burgers equation was tested with a set of $N_u = 100$ randomly distributed initial and boundary data, and the model was trained 3021 parameters of a 9-layer deep neural network using the mean squared error loss to learn the latent solution $u(t, x)$. Each hidden layer contained 20 neurons and a hyperbolic tangent activation function In general, the neural network should be given sufficient approximation capacity in order to accommodate the anticipated complexity of $u(t, x)$. However, in this example, our choice aims to highlight the robustness of the proposed method with respect to the well known issue of over-fitting. Specifically, the term in MSE_f in equation acts as a regularization mechanism that penalizes solutions that do not satisfy equation. Therefore, a key property of physics informed neural networks is that they can be effectively trained using small data sets; a setting often encountered in the study of physical systems for which the cost of data acquisition may be prohibitive.

The exact solution for this problem is analytically available, and the resulting prediction error is measured at $6.7 \cdot 10^{-4}$ in the relative L_2-norm.

CHAPTER 4

Conclusion

The Physics Informed Neural Networks (PINNs) are the deep learning models (here MLP) with certain physics equations injected as a penalizing term to solve various equations, that are equipped with very small data. The ultimate use of a PINN is convergence to solution much faster that any other deep learning models, even in case of equations that does not have a theoretical solution existing. This work explores solving of Burger's equation and Navier Stokes equation. Navier Stokes equation does not have a theoretical solution existing but even then PINN obtained a approximate solution. Whereas Burger's equation was trained and solved to check the accuracy and performance of the model with small data set, so that it could be used for solving complex equations.

The papers by Maziar Raissi, Paris Perdikaris, and George Em Karniadakis [1] [2] has laid a foundation for physics informed neural networks. The first paper explains how the hidden latent solution predicts the solution for various physics equation that models involving advection-diffusion or flow of a viscous liquid using nonlinear partial differential equations. The second paper summarizes to find optimum λ values that will result in finding accurate solutions for the physics equation. Both the papers intend to arrive at a solution with a small data by injecting some some physics information to the deep learning model.

Initial idea about a deep learning model would be to use it for large data set, but these papers introduce to solve an equation with small data set. The Possibility of the approach is because of the involvement of basic knowledge of various equations by the model itself. This could viewed as a new approach to solve various complex equations without performing deductions from the original equation but by making a model predict the right answer.

BIBLIOGRAPHY

[1] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, “Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations”, *arXiv:1711.10566v1*, 2017.

[2] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis, “Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations”, *preprint, arXiv:1711.10566v1,* 2017.

[3] Maziar Raissi and George EmKarniadakis, “Hidden physics models: Machine learning of nonlinear partial differential equations”, *Journal of Computational Physics,* vol. 357, pp. 125 - 141, 2018.