Project Proposals

Damien Beecroft

August 2022

1 Combining Numerics and PINNs for Optimal Modelling of Differential Equations

1.1 Abstract

How does one appropriately combine the vast wealth of theory from numerical analysis and dynamical systems with physics informed neural networks to most efficiently and accurately solve differential equations? This is a complex problem whose answer depends heavily upon the equation, the available data, and a multitude of other factors. We will explore this trade-off to provide guidelines for how to incorporate data and theory effectively for a range of different scenarios through a variety of test problems with wildly different properties: the heat equation, the Helmholtz equation, Burger's equation, and Navier-Stokes equation.

1.2 Introduction

In this paper we study the efficacy of numerics assisted PINNs. In this work a numerics assisted PINN refers to an algorithm where a numerical method that serves as the low fidelity source to a multifidelity physics informed neural network (MFPINN) [2]. This framework is a natural way to incorporate numerical methods and PINNs. Furthermore, other methods (such as training a PINN to learn the residual of a numerical solution) can be interpreted as a sub-case of training a physics assisted PINN. There are three broad metrics for success in the trials that we perform in this paper: accuracy, speed, and generalizability. With these three variables we will construct the Pareto front of numerics assister PINNs as one shifts the computational burden between the numerical method and the MFPINN. This will give the reader a holistic view of the trade-offs between numerics and machine learning in physics applications.

1.3 Body

Accuracy and Speed

From the studies of Kochkov et al. [1] we know that there exist scenarios in which machine learning indeed extends the Pareto front of PDE solvers. Kochkov et al. [1] examines the usage of machine learning to improve finite volume method approximations of fluids with high accuracy solution data. They report that their method is roughly eighty times faster than the classical finite volume method. Furthermore, their model is trained on local information and can therefore be scaled to be used for different initial and boundary conditions. We will reproduce these results within these studies and subsequently extend upon them to see how the performance changes as one has less solution data and must rely more and more on the physical constraints.

Incorporating a numerical method into the PINN framework will help overcome convergence issues that hinder the efficacy of PINNs. Numerical methods do not have a spectral bias that makes it difficult to converge to oscillatory solutions [4]. Furthermore, numerical algorithms do not converge to unstable fixed points as PINNs often do [3]. There are many ways that numerics can correct pathological behavior in PINNs. It remains to be seen whether the converse also occurs. Can PINNs correct for the biases of numerical methods? We will use simulations of finite volume methods on Burger's equation to tackle this problem. Certain finite volume methods introduce artificial viscosity into the solution due to approximation error. We will examine whether PINNs can detect that this phenomena is non-physical and subsequently correct for it.

The experiments will be performed as follows. For each differential equation we will choose a high accuracy numerical method and use it as the true solution. Then, we will vary the amount of computational resouces of the numerical method and the MFPINN. For instance, say that I have a set of grid sizes G and a set of MFPINN sizes N. We will run numerics assisted PINNs on every combination of grid size in G and network size in N. In otherwords, we will sample our numerics assisted PINN architecture from $G \times N$. We will run each model until it has achieved the accuracy of the "true solution" is reached or the method has hit a preset training time. These studies will be repeated for each problem with a varying amount of solutions data. We will also perform extensive studies on problems with irregular domains since this has proven to be a major weakness of numerical methods. With all this information we hope to get a holistic view of when PINNs can improve upon classical numerical schemes.

Generalizability

Another important aspect of our tests is generalizability. It may be worth training a model for a long time given that it can solve a range of problems without retraining. We will take the numerics assisted PINNs trained in the last section and see how well they perform on the same differential equations with different initial and boundary conditions. We will plot how the accuracy of the numerics assisted PINNs changes depending on how different the initial and boundary conditions are from any of the problems in the training data set. We expect differential equations with predominantly local dynamics (Burger's equation and Navier Stokes) to have an easier time generalizing than differential equations whose local dynamics are strongly coupled to the global state (the Helmholtz and heat equations).

1.4 Expected Takeaways and "Who Cares?"

The field of physics informed machine learning has serious problems with reproducibility and benchmarking. Quality of PINN results are often dependent on hyperparameters that are tuned through an arduous cycle of trial and error. This hyperparameter tuning is fine and good for showing how far one can push an algorithm. However, the need for tuning does not bode well for the robustness of PINNs. In my studies I intend to factor in the time required for hyperparameter tuning into the comparison so that we may ascertain the strength of the method.

Adding numerics to the PINN training process undoubtedly makes convergence easier. If a PINN is incapable of improving on a numerical method in this simpler scenario it is a sign that machine learning may not an effective tool for this particular problem. Hopefully we will be able to extract patterns from these experiments and come up with guiding principles that can help engineers make informed decisions on how to effectively combine the vast wealth of theory from numerical analysis and dynamical systems with physics informed neural networks.

2 Analysis of Mini-Batch Gradient Descent for Physics Informed Neural Nets

2.1 Abstract

The convergence rates of numerical methods are heavily dependent on the density of collocation points in the solution domain. Physics informed neural networks (PINNs) attempt to solve the same problems, however there are no theoretical results that provide guidance as to the quantity and density of collocation points in each batch to achieve convergence. This is the topic of this paper.

2.2 Introduction

Suppose that we are attempting to solve the following evolution equation on some open solution domain Ω with boundary $\partial\Omega$ for time $t \in (0, t_{\mathrm{f}}] = T$. $\Omega, \partial\Omega \subset \mathbb{R}^d$.

$$\begin{aligned} u^*(t,x)_t + N[u^*(t,x)] = 0 &\quad \text{for} \quad x \in \Omega,\ t \in T \\ B[u^*(t,x)] = 0 &\quad \text{for} \quad x \in \partial\Omega,\ t \in T \\ u^*(0,x) - u_0(x) = 0 &\quad \text{for} \quad x \in \Omega \cup \partial\Omega \end{aligned} \tag{1}$$

In equation 2.2 N and B are differential operators and $u_0(x)$ is the initial state of the differential equation. We assume throughout this report that this differential equation is well posed with $u^* : \mathbb{R}^d \to \mathbb{R}^p$ as the true solution. Henceforth, we refer to $\Omega \cup \partial\Omega$ as $\overline{\Omega}$.

We want to train a neural network to solve equation 2.2. We call this neural network $u(t, x; \theta)$. θ denotes the parameters of the neural net. We often drop the inputs or parameters of $u(t, x; \theta)$ for simplicity of notation, opting to call it $u(t, x)$ or u instead. We would like this network to achieve a low loss in the following sense.[1]

$$\mathcal{L}_{\sup} = \sup_{(t,x)\in T\times\Omega} \|u_t + N[u]\|_2^2 + \sup_{(t,x)\in T\times\partial\Omega} \|B[u]\|_2^2 + \sup_{(t,x)\in\{0\}\times\overline{\Omega}} \|u - u_0\|_2^2 \tag{2}$$

Unfortunately, the above loss is not feasible to optimize with gradient descent since it is not differentiable. We therefore turn our attention to a surrogate loss.

$$\mathcal{L}_{\mathrm{int}} = \int_T \int_\Omega \|u_t + N[u]\|_2^2 \mathrm{d}x\mathrm{d}t + \int_T \int_{\partial\Omega} \|B[u]\|_2^2 \mathrm{d}x\mathrm{d}t + \int_{\overline{\Omega}} \|u(0,x) - u_0(x)\|_2^2 \mathrm{d}x \tag{3}$$

This loss function is differentiable.[2] However, these integrals cannot be computed analytically. In practice researchers sample collocation points separately from the sets $T \times \Omega$, $T \times \partial\Omega$, and $\{0\} \times \overline{\Omega}$. We refer to these as the solution, boundary, and initial domains respectively. These mini-batches of collocation points are used to update the neural network parameters, θ. For now, let us assume that the points are sampled uniformly at random from each of domains. The process we have just described culminates in the following algorithm.

In this paper we study when we can expect algorithm 1 to converge in the sense of $\mathcal{L}_{\mathrm{int}}$ and therefore (I believe) $\mathcal{L}_{\sup}$.

[1]For now we ignore the regularization of the network through parameter magnitude penalization and the incorporation of solution data into the model. I will probably revisit the problem with these considerations later on.

[2]I believe that I can bound $\mathcal{L}_{\sup}$ in terms of $\mathcal{L}_{\mathrm{int}}$ assuming that our objective function is Lipschitz. If I can do this I can achieve a type of uniform convergence of the solution which would be very nice.

Algorithm 1 Physics Informed Neural Network Training Process

```
Input: Neural network–$u(t, x; \theta)$, Number of training iteration–M, Learning rate–$\epsilon$
for j in 1:M do
    Sample $\{(t_j^s, x_j^s)\}_{j=1}^{N_s}$ where $(t_k^s, x_k^s) \sim \text{Unif}(T \times \Omega)$    ▷ Sample solution domain collocation points
    Sample $\{(t_j^b, x_j^b)\}_{j=1}^{N_b}$ where $(t_k^b, x_k^b) \sim \text{Unif}(T \times \partial\Omega)$    ▷ Sample boundary domain collocation points
    Sample $\{(t_j^i, x_j^i)\}_{j=1}^{N_i}$ where $(t_k^i, x_k^i) \sim \text{Unif}(\{0\} \times \overline{\Omega})$    ▷ Sample initial domain collocation points
    $L_s \leftarrow \sum_{j=1}^{N_s} \|u(t_j^s, x_j^s)_t + N[u(t_j^s, x_j^s)]\|_2^2$
    $L_b \leftarrow \sum_{j=1}^{N_b} \|B[u(t_j^b, x_j^b)]\|_2^2$
    $L_i \leftarrow \sum_{j=1}^{N_i} \|u(t_j^i, x_j^i) - u_0(x_j^i)\|_2^2$
    $L \leftarrow L_b + L_s + L_i$
    $\theta \leftarrow \theta - \epsilon \nabla_\theta L$
end for
Return: $u(t, x; \theta)$
```

Project Two

After the PINN experiments are done I want to repeat the studies from project one with multifidelity finite basis DeepONets. DeepONets do not work well alone. I wonder whether the aide of numerical analysis can make DeepONets viable. This is the very important experiment. Having to retrain a network for every set of initial conditions is unreasonable in a wide range of applications.

Project Three

I think it would be interesting to look at the loss landscape of the MFFBPINNs with numerical solvers. We know that at the beginning of training you are within a certain error ϵ of the true solution. This means that evaluating a linear or quadratic approximation of the neural network at initialization would likely give one a good approximation of the loss landscape at the minima. Furthermore, there is something that has struck me as a bit strange about the PINN training process. In numerical methods the density of grid points is paramount to successful convergence. However, density of collocation points in batches for PINNs is not–at least in my experience–treated with the same importance. When one creates a batch of collocation points I believe they should be sampled at or above the Nyquist frequency of the highest frequency mode in the solution. I am curious to see whether my intuition can be tested by analyzing the loss landscape. It may also be interesting to analyze the convergence of MFFBPINNs with numerical solvers through a combination of numerical solvers and the neural tangent kernel.

References

[1] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[2] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *Journal of Computational Physics*, 401:109020, jan 2020.

[3] Franz M. Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C. Geiger. On the role of fixed points of dynamical systems in training physics-informed neural networks, 2023.

[4] Zhi-Qin John Xu. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767, jun 2020.