Title: Physics-informed neural networks

URL: https://en.wikipedia.org/wiki/Physics-informed_neural_networks

PageID: 67944516

Categories: Category:Deep learning, Category:Differential equations

-----

Physics-informed neural networks (PINNs), [1] also referred to as Theory-Trained Neural Networks (TTNs), [2] are a type of universal function approximators that can embed the knowledge of any physical laws that govern a given data-set in the learning process, and can be described by partial differential equations (PDEs). Low data availability for some biological and engineering problems limit the robustness of conventional machine learning models used for these applications. [1] The prior knowledge of general physical laws acts in the training of neural networks (NNs) as a regularization agent that limits the space of admissible solutions, increasing the generalizability of the function approximation. This way, embedding this prior information into a neural network results in enhancing the information content of the available data, facilitating the learning algorithm to capture the right solution and to generalize well even with a low amount of training examples. For they process continuous spatial and time coordinates and output continuous PDE solutions, they can be categorized as neural fields .

Function approximation

Most of the physical laws that govern the dynamics of a system can be described by partial differential equations. For example, the Navier–Stokes equations [3] are a set of partial differential equations derived from the conservation laws (i.e., conservation of mass , momentum , and energy) that govern fluid mechanics . The solution of the Navier–Stokes equations with appropriate initial and boundary conditions allows the quantification of flow dynamics in a precisely defined geometry. However, these equations cannot be solved exactly and therefore numerical methods must be used (such as finite differences , finite elements and finite volumes). In this setting, these governing equations must be solved while accounting for prior assumptions, linearization, and adequate time and space discretization.

Recently, solving the governing partial differential equations of physical phenomena using deep learning has emerged as a new field of scientific machine learning (SciML), leveraging the universal approximation theorem [4] and high expressivity of neural networks. In general, deep neural networks could approximate any high-dimensional function given that sufficient training data are supplied. [5] However, such networks do not consider the physical characteristics underlying the problem, and the level of approximation accuracy provided by them is still heavily dependent on careful specifications of the problem geometry as well as the initial and boundary conditions. Without this preliminary information, the solution is not unique and may lose physical correctness. On the other hand, physics-informed neural networks (PINNs) leverage governing physical equations in neural network training. Namely, PINNs are designed to be trained to satisfy the given training data as well as the imposed governing equations. In this fashion, a neural network can be guided with training data that do not necessarily need to be large and complete. [5] Potentially, an accurate solution of partial differential equations can be found without knowing the boundary conditions. [6] Therefore, with some knowledge about the physical characteristics of the problem and some form of training data (even sparse and incomplete), PINN may be used for finding an optimal solution with high fidelity.

PINNs allow for addressing a wide range of problems in computational science and represent a pioneering technology leading to the development of new classes of numerical solvers for PDEs. PINNs can be thought of as a meshfree alternative to traditional approaches (e.g., CFD for fluid dynamics), and new data-driven approaches for model inversion and system identification. [7] Notably, the trained PINN network can be used for predicting the values on simulation grids of different resolutions without the need to be retrained. [8] In addition, being neural fields, they allow for exploiting automatic differentiation (AD) [9] to compute the required derivatives in the partial differential equations, a new class of differentiation techniques widely used to derive neural networks assessed to be superior to numerical or symbolic differentiation .

Modeling and computation

A general nonlinear partial differential equation can be:

u t + N [u ; λ] = 0 , x ∈ Ω , t ∈ [0 , T] {\displaystyle u_{t}+N[u;\lambda]=0,\quad x\in \Omega ,\quad t\in [0,T]}

where u (t , x) {\displaystyle u(t,x)} denotes the solution, N [⋅ ; λ] {\displaystyle N[\cdot ;\lambda]} is a nonlinear operator parameterized by λ {\displaystyle \lambda } , and Ω {\displaystyle \Omega } is a subset of R D {\displaystyle \mathbb {R} ^{D}} . This general form of governing equations summarizes a wide range of problems in mathematical physics, such as conservative laws, diffusion process, advection-diffusion systems, and kinetic equations. Given noisy measurements of a generic dynamic system described by the equation above, PINNs can be designed to solve two classes of problems:

data-driven solution

data-driven discovery of partial differential equations.

Data-driven solution of partial differential equations

The data-driven solution of PDE [1] computes the hidden state u (t , x) {\displaystyle u(t,x)} of the system given boundary data and/or measurements z {\displaystyle z} , and fixed model parameters λ {\displaystyle \lambda } . We solve:

u t + N [u] = 0 , x ∈ Ω , t ∈ [0 , T] {\displaystyle u_{t}+N[u]=0,\quad x\in \Omega ,\quad t\in [0,T]} .

By defining the residual f (t , x) {\displaystyle f(t,x)} as

f := u t + N [u] {\displaystyle f:=u_{t}+N[u]} ,

and approximating u (t , x) {\displaystyle u(t,x)} by a deep neural network. This network can be differentiated using automatic differentiation. The parameters of u (t , x) {\displaystyle u(t,x)} and f (t , x) {\displaystyle f(t,x)} can be then learned by minimizing the following loss function L t o t {\displaystyle L_{tot}} :

L t o t = L u + L f {\displaystyle L_{tot}=L_{u}+L_{f}} .

Where L u = ■ u − z ■ Γ {\displaystyle L_{u}=\Vert u-z\Vert _{\Gamma }} is the error between the PINN u (t , x) {\displaystyle u(t,x)} and the set of boundary conditions and measured data on the set of points Γ {\displaystyle \Gamma } where the boundary conditions and data are defined, and L f = ■ f ■ Γ {\displaystyle L_{f}=\Vert f\Vert _{\Gamma }} is the mean-squared error of the residual function. This second term encourages the PINN to learn the structural information expressed by the partial differential equation during the training process.

This approach has been used to yield computationally efficient physics-informed surrogate models with applications in the forecasting of physical processes, model predictive control, multi-physics and multi-scale modeling, and simulation. [10] It has been shown to converge to the solution of the PDE. [11]

Data-driven discovery of partial differential equations

Given noisy and incomplete measurements z {\displaystyle z} of the state of the system, the data-driven discovery of PDE [7] results in computing the unknown state u (t , x) {\displaystyle u(t,x)} and learning model parameters λ {\displaystyle \lambda } that best describe the observed data and it reads as follows:

u t + N [u ; λ] = 0 , x ∈ Ω , t ∈ [0 , T] {\displaystyle u_{t}+N[u;\lambda]=0,\quad x\in \Omega ,\quad t\in [0,T]} .

By defining f (t , x) {\displaystyle f(t,x)} as

f := u t + N [u ; λ] = 0 {\displaystyle f:=u_{t}+N[u;\lambda]=0} , and approximating u (t , x) {\displaystyle u(t,x)} by a deep neural network, f (t , x) {\displaystyle f(t,x)} results in a PINN. This network can be derived using automatic differentiation. The parameters of u (t , x) {\displaystyle u(t,x)} and f (t , x) {\displaystyle f(t,x)} , together with the parameter λ {\displaystyle \lambda } of the differential operator can be then learned by minimizing the following loss function L t o t {\displaystyle L_{tot}} :

L t o t = L u + L f {\displaystyle L_{tot}=L_{u}+L_{f}} .

Where L u = ■ u − z ■ Γ {\displaystyle L_{u}=\Vert u-z\Vert _{\Gamma }} , with u {\displaystyle u} and z {\displaystyle z} state solutions and measurements at sparse location Γ {\displaystyle \Gamma } , respectively and L f = ■ f ■ Γ {\displaystyle L_{f}=\Vert f\Vert _{\Gamma }} residual function. This second term requires the structured information represented by the partial differential equations to be satisfied in the training process.

This strategy allows for discovering dynamic models described by nonlinear PDEs assembling computationally efficient and fully differentiable surrogate models that may find application in predictive forecasting, control, and data assimilation . [12] [13] [14] [15] [16]

Physics-informed neural networks for piece-wise function approximation

PINN is unable to approximate PDEs that have strong non-linearity or sharp gradients that commonly occur in practical fluid flow problems. Piece-wise approximation has been an old practice in the field of numerical approximation. With the capability of approximating strong non-linearity extremely light weight PINNs are used to solve PDEs in much larger discrete subdomains that increases accuracy substantially and decreases computational load as well. [17] [18] DPINN (Distributed physics-informed neural networks) and DPIELM (Distributed physics-informed extreme learning machines) are generalizable space-time domain discretization for better approximation. [17] DPIELM is an extremely fast and lightweight approximator with competitive accuracy. Domain scaling on the top has a special effect. [18] Another school of thought is discretization for parallel computation to leverage usage of available computational resources.

XPINNs [19] is a generalized space-time domain decomposition approach for the physics-informed neural networks (PINNs) to solve nonlinear partial differential equations on arbitrary complex-geometry domains. The XPINNs further pushes the boundaries of both PINNs as well as Conservative PINNs (cPINNs), [20] which is a spatial domain decomposition approach in the PINN framework tailored to conservation laws. Compared to PINN, the XPINN method has large representation and parallelization capacity due to the inherent property of deployment of multiple neural networks in the smaller subdomains. Unlike cPINN, XPINN can be extended to any type of PDEs. Moreover, the domain can be decomposed in any arbitrary way (in space and time), which is not possible in cPINN. Thus, XPINN offers both space and time parallelization, thereby reducing the training cost more effectively. The XPINN is particularly effective for the large-scale problems (involving large data set) as well as for the high-dimensional problems where single network based PINN is not adequate. The rigorous bounds on the errors resulting from the approximation of the nonlinear PDEs (incompressible Navier–Stokes equations) with PINNs and XPINNs are proved. [15] However, DPINN debunks the use of residual (flux) matching at the domain interfaces as they hardly seem to improve the optimization. [18]

Physics-informed neural networks and theory of functional connections

In the PINN framework, initial and boundary conditions are not analytically satisfied, thus they need to be included in the loss function of the network to be simultaneously learned with the differential equation (DE) unknown functions. Having competing objectives during the network's training can lead to unbalanced gradients while using gradient-based techniques, which causes PINNs to often struggle to accurately learn the underlying DE solution. This drawback is overcome by using functional interpolation techniques such as the Theory of functional connections (TFC)'s constrained expression, in the Deep-TFC [21] framework, which reduces the solution search space of constrained problems to the subspace of neural network that analytically satisfies the constraints. [22] A further improvement of PINN and functional interpolation approach is given by the Extreme Theory of Functional Connections (X-TFC) framework, where a single-layer Neural Network and the extreme learning machine training algorithm are employed. [23] X-TFC allows to improve the accuracy and performance of regular PINNs, and its robustness and reliability are proved for stiff problems, optimal control, aerospace, and rarefied gas dynamics applications. [24] [25] [26]

Physics-informed PointNet (PIPN) for multiple sets of irregular geometries

Regular PINNs are only able to obtain the solution of a forward or inverse problem on a single geometry. It means that for any new geometry (computational domain), one must retrain a PINN. This limitation of regular PINNs imposes high computational costs, specifically for a comprehensive investigation of geometric parameters in industrial designs. Physics-informed PointNet (PIPN) [27] is fundamentally the result of a combination of PINN's loss function with PointNet. [28] In fact, instead of using a simple fully connected neural network, PIPN uses PointNet as the core of its neural network. PointNet has been primarily designed for deep learning of 3D object classification and segmentation by the research group of Leonidas J. Guibas . PointNet extracts geometric features of input computational domains in PIPN. Thus, PIPN is able to solve governing equations on multiple computational domains (rather than only a single domain) with irregular geometries, simultaneously. The effectiveness of PIPN has been shown for incompressible flow , heat transfer and linear elasticity . [27] [29]

Physics-informed neural networks (PINNs) for inverse computations

Physics-informed neural networks (PINNs) have proven particularly effective in solving inverse problems within differential equations, [30] demonstrating their applicability across science, engineering, and economics. They have shown to be useful for solving inverse problems in a variety of fields, including nano-optics, [31] topology optimization/characterization, [32] multiphase flow in porous media, [33] [34] and high-speed fluid flow. [35] [13] PINNs have demonstrated flexibility when dealing with noisy and uncertain observation datasets. They also demonstrated clear advantages in the inverse calculation of parameters for multi-fidelity datasets, meaning datasets with different quality, quantity, and types of observations. Uncertainties in calculations can be evaluated using ensemble-based or Bayesian-based calculations. [36] [37]

PINNs can also be used in connection with symbolic regression for discovering the mathematical expression in connection with discovery of parameters and functions. One example of such application is the study on chemical ageing of cellulose insulation material, [38] in this example PINNs are used to first discover a parameter for a set of ordinary differential equations (ODEs) and later a function solution, which is later used to find a more fitting expression using a symbolic regression with a combination of operators.

Physics-informed neural networks for elasticity problems

Ensemble of physics-informed neural networks is applied for solving plane elasticity problems. Surrogate networks are intended for the unknown functions, namely, the components of the strain and the stress tensors as well as the unknown displacement field, respectively. The residual network provides the residuals of the partial differential equations (PDEs) and of the boundary conditions. The computational approach is based on principles of artificial intelligence. [39] This approach can be extended to nonlinear elasticity problems, where the constitutive equations are nonlinear. PINNs can also be used for Kirchhoff plate bending problems with transverse distributed loads and to contact models with elastic Winkler’s foundations. [40]

Physics-informed neural networks (PINNs) with backward stochastic differential equation

Deep backward stochastic differential equation method is a numerical method that combines deep learning with Backward stochastic differential equation (BSDE) to solve high-dimensional problems in financial mathematics. By leveraging the powerful function approximation capabilities of deep neural networks , deep BSDE addresses the computational challenges faced by traditional numerical methods like finite difference methods or Monte Carlo simulations, which struggle with the curse of dimensionality. Deep BSDE methods use neural networks to approximate solutions of high-dimensional partial differential equations (PDEs), effectively reducing the computational burden. Additionally, integrating Physics-informed neural networks (PINNs) into the deep BSDE framework enhances its capability by embedding the underlying physical laws into the neural network architecture, ensuring solutions adhere to governing stochastic differential equations, resulting in more accurate and reliable solutions. [41]

Physics-informed neural networks for biology

An extension or adaptation of PINNs are Biologically-informed neural networks (BINNs). BINNs introduce two key adaptations to the typical PINN framework: (i) the mechanistic terms of the governing PDE are replaced by neural networks, and (ii) the loss function L_{tot} is modified to include L_{constr}, a term used to incorporate domain-specific knowledge that helps enforce biological applicability. For (i), this adaptation has the advantage of relaxing the need to specify the governing differential equation a priori, either explicitly or by using a library of candidate terms. Additionally, this approach circumvents the potential issue of misspecifying regularization terms in stricter theory-informed cases. [42] [43]

A natural example of BINNs can be found in cell dynamics, where the cell density $u(x,t)$ is governed by a reaction-diffusion equation with diffusion and growth functions $D(u)$ and $G(u)$, respectively:

$$u_{t}=\nabla \cdot (D(u)\nabla u)+G(u)u,\quad x\in \Omega ,\quad t\in [0,T]$$

In this case, a component of L_{constr} could be $||D||_{\Gamma }$ for $D < D_{min}, D > D_{max}$, which penalizes values of D that fall outside a biologically relevant diffusion range defined by $D_{min}\leq D\leq D_{max}$. Furthermore, the BINN architecture, when utilizing multilayer-perceptrons (MLPs), would function as follows: an MLP is used to construct $u_{MLP}(x,t)$ from model inputs (x,t), serving as a surrogate model for the cell density $u(x,t)$. This surrogate is then fed into the two additional MLPs, $D_{MLP}(u_{MLP})$ and $G_{MLP}(u_{MLP})$, which model the diffusion and growth functions. Automatic differentiation can then be applied to compute the necessary derivatives of u_{MLP}, D_{MLP} and G_{MLP} to form the governing reaction-diffusion equation. [42]

Note that since u_{MLP} is a surrogate for the cell density, it may contain errors, particularly in regions where the PDE is not fully satisfied. Therefore, the reaction-diffusion equation may be solved numerically, for instance using a method-of-lines approach .

Limitations

Translation and discontinuous behavior are hard to approximate using PINNs. [18] They fail when solving differential equations with slight advective dominance and hence asymptotic behaviour causes the method to fail. Such PDEs could be solved by scaling variables. [18] This difficulty in training of PINNs in advection-dominated PDEs can be explained by the Kolmogorov n–width of the solution. [44] They also fail to solve a system of dynamical systems and hence have not been a success in solving chaotic equations. [45] [46] One of the reasons behind the failure of regular PINNs is soft-constraining of Dirichlet and Neumann boundary conditions which pose a multi-objective optimization problem which requires manually weighing the loss terms to be able to optimize. [18] [47] More generally, posing the solution of a PDE as an optimization problem brings with it all the problems that are faced in the world of optimization, the major one being getting stuck in local optima. [18] [48] Specifically, it was shown that some of these local optima are connected to unstable solutions of a given PDE. [49]

References

External links

Physics Informed Neural Network

PINN – repository to implement physics-informed neural network in Python

XPINN – repository to implement extended physics-informed neural network (XPINN) in Python

PIPN [2] – repository to implement physics-informed PointNet (PIPN) in Python