1. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

project codes (https://github.com/maziarraissi/PINNs)

@article{raissi2019physics, title={Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations}, author={Raissi, Maziar and Perdikaris, Paris and Karniadakis, George E}, journal={Journal of Computational Physics}, volume={378}, pages={686--707}, year={2019}, publisher={Elsevier} }

This paper concentrates in parametrized and nonlinear pde of the form

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

where $u(t, x)$ denotes the latent solution, $\mathcal{N}[. ; \lambda]$ is a nonlinear operator parametrized by λ, Ω is a subset of $\mathbb{R}^D$.

Continuous time model

Let $f := u_t + \mathcal{N}[u; \lambda]$ then the authors studies the shared parameters between the two neural networks $u(t, x)$ and $f(t, x)$ by minimizing the mean squared error loss

$$MSE = MSE_u + MSE_f$$

where

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

and

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

Here, $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t, x)$ and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ 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 the pde at a finite set of collocation points.

Possible limitation:

- Need to use a large number of collocation points N_f in order to enforce physics-informed contraints in the entire spatio-temporal domain --> dimensional curse issue for high-dimensional problems. This can be addressed to some extend using *sparse grid or Monte-Carlo samling*, using a more structured neural network representation leverageing the classical Runge-Kutta time-stepping schemes in the next section of the paper.

Discrete time model

The authors apply a q-stages to $u_t + \mathcal{N}[u;\lambda] = 0, x \in \Omega, t \in [0,T]$ and obtain

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

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

where $u^{n+c_j}(x) = u(t^n + c_j\Delta t, x)$ for $j = 1,\dots,q$. So, $u^{n+c_j}(x)$ is the hidden state of the system at time $t^n + c_j\Delta t$ for $j = 1,\dots,q$. This system is equivalent to

$$u^n = u_i^n, i = 1,..,q,$$

$$u^n = u_{q+1}^n,$$

where

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

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

Then, they place a multi-output neural network prior on

$$u^{n+c_1}(x),\dots,u^{n+c_q}(x), u^{n+1}(x)$$

So, the input is x and outputs are

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

The paper also presents an approach to data-driven discovery pde similar to the 2 previous scheme for solving pde.

Questions:

- How deep/wide the network should be?
- How much data is needed?
- Effect of activation functions?
- Are the MSE and sum of squared errors the appropriate functions?

DGM: A deep learning algorithm for solving partial differential equations

@article{sirignano2018dgm, title={DGM: A deep learning algorithm for solving partial differential equations}, author={Sirignano, Justin and Spiliopoulos, Konstantinos}, journal={Journal of Computational Physics}, volume={375}, pages={1339--1364}, year={2018}, publisher={Elsevier} }

Consider the potentially nonlinear pde

$$\partial_t u(t,x) + \mathcal{L}u(t,x) = 0, \quad (t,x) \in [0,T] \times \Omega$$

$$u(0,x) = u_0(x),$$

$$u(t,x) = g(t,x), \quad x \in \partial\Omega$$

The unknown solution $u(t,x)$ is usually approximate by $f(t,x)$ by minimizing the L^2 loss

$$J(f) = ||\partial_t f + \mathcal{L}f||^2_{2,[0,T]\times\Omega} + ||f - g||^2_{2,[0,T]\times\partial\Omega} + ||f(0,.) - u_0||^2_{2,\Omega}.$$

However, in this paper, the authors use a different parametrization. Instead of using $f(t,x)$, the unknown solution $u(t,x)$ is approximated using a deep neural network $f(t,x;\theta), \theta \in \mathbb{R}^K$, where θ are parameters. The objective function is

$$J(f) = ||\partial_t f(t,x;\theta) + \mathcal{L}f(t,x;\theta)||^2_{[0,T]\times\Omega,\nu_1} + ||f(t,x;\theta) - g(t,x)||^2_{[0,T]\times\partial\Omega,\nu_2} + ||f(0,x;\theta) - u_0(x)||^2_{\Omega,\nu_3}.$$

and it is minimized using stochastic gradient descent. Each loop n is trained with a random point (t_n, x_n) is generated from $[0,T] \times \Omega$ w.r.t probability densities ν_1, a random point (τ_n, z_n) is generated from $[0,T] \times \partial\Omega$ w.r.t probability densities ν_2, a random point w_n is generated from Ω w.r.t probability densities ν_3.

Problem: since stochastic gradient descent is used, stability poses an important questions.

Data-driven discovery of partial differential equations

Code (https://github.com/snagcliffs/PDE-FIND)

@article{rudy2017data, title={Data-driven discovery of partial differential equations}, author={Rudy, Samuel H and Brunton, Steven L and Proctor, Joshua L and Kutz, J Nathan}, journal={Science Advances}, volume={3}, number={4}, pages={e1602614}, year={2017}, publisher={American Association for the Advancement of Science} }

In this paper, the author propose a method using ridge regression with hard thresholding to discover the governing pde of a given system. They parameterize the pde of general form

$$u_t = \mathcal{N}(u, u_x, u_{xx}, \ldots, x, \mu)$$

The key assumption is that the function $\mathcal{N}$ consists of only a few terms so that the functional form is relatively sparse. This assumption itself limits the application of this method.

The basic idea is to first collect data $U \in \mathbb{C}^{n\times m}$ of m time points and n spatial locations. We can also collect additional input such as the magnitude of complex data, a potential for Schodinger equation into $Q \in \mathbb{C}^{n\times m}$. Next, a library $\Phi(U,Q) \in C^{nm\times D}$ of candidate terms

$1, u, u^2, u_x, u_{xx}, \ldots$ is built, and are regarded as independent variable. The problem turn into a least square problem, where we try to find the sparse vector ξ of coefficients by minimizing

$$||\Phi(U, Q)\xi - U_t||_2^2 + \epsilon\kappa(\Phi(U, Q))||\xi||_0$$

where $\kappa(\Phi)$ is the condition number of Φ, which indicates regularization against ill-posed problem.

The advantages of this method are:

1. measurements can be collected in either a fixed or moving frame (Eulerian or Lagrangian)
2. efficiently handle high-dimensional data through innovative sampling strategies

Strategies: They use subsampled data. Subsampled data is created by randomly select a set spatial points and uniformly subsample in time. Although only a small fraction of the spatial points are in the sample, nearby points are needed to compute the derivative terms. The derivatives are computed using a small number of local points near each position via polynomial interpolation. This means we uses local information around each measurement in the sample.

Drawbacks associated with sparse regression techniques:

- Derivatives are taken using polynomial interpolation, whose numerical approximations are ill-conditioned and unstable, due to truncation and round-off errors (see more in Baydin (https://arxiv.org/abs/1502.05767)) . Moreover, the evaluation of derivatives requires a large number of data, which makes the data amount of data used is much more than the size of subsampled data, which is used for regression. Thus, this method still requires far more data points than the number of functions in the library.
- The author assume that the chosen library is sufficiently rich to have a sparse representation of the underlying system. However, when the dynamics are unknown, this may not be true. In higher dimensional problems, the number of functions in the library increase exponentially.

Deep hidden physics models: deep learning for nonlinear partial differential equations

Code (https://github.com/maziarraissi/DeepHPMs)

@article{raissi2018deep, title={Deep hidden physics models: Deep learning of nonlinear partial differential equations}, author={Raissi, Maziar}, journal={The Journal of Machine Learning Research}, volume={19}, number={1}, pages={932--955}, year={2018}, publisher={JMLR. org} }

In this paper, the author propose a deep learning approach to discover the governing pde of a given system. They parameterize the pde of general form

$$u_t = \mathcal{N}(u, u_x, u_{xx}, u_{xxx}, \ldots, x, \mu)$$

They define a deep hidden physic model f given by

$$f = u_t - \mathcal{N}(u, u_x, u_{xx}, u_{xxx}, \ldots, x, \mu)$$

The parameters of u and $\mathcal{N}$ can be learned by minimizing the following sum of squared errors

$$SSE := \sum_{i=1}^{N}(|u(t^i, x^i) - u^i|^2 + |f(t^i, x^i)|^2),$$

where $\{t^i, x^i, u^i\}_{i=1}^N$ is training data.

The choice of activation functions, number of layers, number of neuron per layer still is an open question. See Raissi (https://arxiv.org/abs/1711.10561), Rassi II (https://arxiv.org/abs/1711.10566), 3 (https://arxiv.org/abs/1801.01236)

There is also a question of how common techniques such as batch normalization, dropout, L^1, L^2 regulirization could enhance the robustness of the algorithm.

Entrée [0]: