Physics Informed Neural Networks

also called Physics-guided or Physics-aware NNs

Aryan Ritwajeet Jha

What problems do we hope to counter using PINNs?

- Because NNs (by NN I always mean Deep Neural Networks or DNNs) typically tend to have the problems given below, when applied to solve physical systems such as Power Systems:
  - Low interpretability in data processing
  - High quantity and quality of required training data
  - Production of physically infeasible/incoherent data
  - No theoretical guarantee on outputs.
  - Not generalizable

Recall: Why do we even want to use any kind of NN?

Why use some black box model when we have more tangible gradient descent iterative models or sometimes even analytical models

- System cannot be feasibly solved for using standard computational tools.
- Conventional solution *too slow*.
- NNs are fast.
- With the right architecture and training data, NNs can solve for (generate outputs for) most real-world system problems.

  “Multilayer feed-forward networks are universal approximators” – 1989 highly cited paper by White et al.

What are PINNs?

- Basically some form of incorporation of real world tried and tested equations into a Deep Neural Network. (I’ll just call DNNs as NNs).

What are PINNs? - **Wang (2023)**

- Four Types/Implementations of PINNs:
  - **Physics-Informed Loss Functions**
  - Physics-Informed Initialization
  - Physics-Informed Design of Architecture
  - Hybrid Physics-Deep Learning Models

What are PINNs?
Physics-Informed Loss Functions

- A neural network whose loss function includes some **Physics-based or Mathematics-based loss function(s)**. These loss functions should be relevant to the system of our concern. We may also call them as *System Error Equations* or *System Loss Functions* or simply *System Equations*. In one paper they call it *Physical Regularization Term* (PRT).

$$\mathcal{L} = L(\hat{\mathbf{y}}, \mathbf{y}) + \lambda R(\mathbf{W}, \mathbf{b}) + \gamma R_{\text{Phy}}(\mathbf{X}, \hat{y})$$

Regular LSE Loss Function

Weight/bias-arresting function

Errors in some *System Equations*.

Recall: Appending the Original Loss Function with more terms is NOT a new thing.

- We already do this in L1 and L2 regressions, where we add an additional loss term proportional to the absolute values or squared values of the NN weights, for reasons.
  - L1: For the sake of removing redundant features
  - L2: For preventing apparently weights of *apparently important* features from ballooning forever.

$$\mathcal{L} = L(\hat{\mathbf{y}}, \mathbf{y}) + \lambda R(\mathbf{W}, \mathbf{b}) + \gamma R_{\text{Phy}}(\mathbf{X}, \hat{y})$$

Regular LSE Loss Function

Weight/bias-arresting function

Errors in some *System Equations*.

Raissi et al's usage: Schrodinger's Equation

Schrödinger equation along with periodic boundary conditions is given by

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$

$$h(0, x) = 2\,\mathrm{sech}(x),$$

$$h(t, -5) = h(t, 5),$$

$$h_x(t, -5) = h_x(t, 5),$$

where $h(t, x)$ is the complex-valued solution. Let us define $f(t, x)$ to be given by

$$f := ih_t + 0.5h_{xx} + |h|^2 h,$$

Raissi et al's usage: Schrodinger's Equation

$$MSE = MSE_0 + MSE_b + MSE_f,$$

where

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2,$$

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b} \left(|h^i(t_b^i, -5) - h^i(t_b^i, 5)|^2 + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2 \right),$$

and

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

- Two additional kinds of loss functions are used: One to enforce boundary conditions and another to penalize against violation of Schrodinger's Equation at the data points.

Raissi et al's usage: Schrodinger's Equation

- 5 Layer NN
- 100 Neurons per hidden layer
- The authors say that a key feature of PINN is how the usage of additional regularization equations such as MSE_b and MSE_f in the Loss function, helps us get 'extract' more quality information from the same dataset, compared to a vanilla NN.
- In other words, **PINNs are more sample efficient**.

Potential Advantages of using PINNs

compared to NNs

A. A more accurate model (objectively quantifiable)

B. Improved Interpretability (?)

C. Physically consistent results

D. *System Equations* can also function as good Bad Data detectors.

Potential Advantages of using PINNs

compared to NNs

D Shrunk search space of the weights

- improved training efficiency (?)
- Better convergence performance (?)

E Enhanced Sampling Efficiency

- Reduced reliance on a large amount of training data

Potential Advantages of using PINNs

compared to NNs

- Better *generalizability*
  - An NN having good *Generalizability* means:
    - It has similar prediction accuracy irrespective of chosen set of training data and testing data
    - Its prediction accuracy for both training data and testing data is about the same.
    - Too complicated NN would overfit training data (‘memorize’ it) and perform poorly on testing data
    - Too simple NN would be unable to handle the complexity (edge test cases) of the system of concern

What do the references say? - **Karniadakis (2019)**

".. for many cases pertaining to the modeling of physical and biological systems, there exists a vast amount of prior knowledge that is currently not being utilized in modern machine learning practice. Let it be the principled physical laws that govern the time-dependent dynamics of a system, or some empirically validated rules or other domain expertise, this prior information can act as a regularization agent that constrains the space of admissible solutions to a manageable size"

- *"We must note however that the proposed methods should not be viewed as replacements of classical numerical methods for solving partial differential equations (e.g., finite elements, spectral methods, etc.)"*
- *" .. classical methods such as the Runge–Kutta time-stepping schemes can coexist in harmony with deep neural networks"*

What do the references say? - **Karniadakis (2019)**

Although a series of promising results was presented, the reader may perhaps agree this work creates more questions than it answers.

- *How deep/wide should the neural network be?*
- *How much data is really needed?*
- *Why does the algorithm converge to unique values for the parameters of the differential operators, i.e., why is the algorithm not suffering from local optima for the parameters of the differential operator?*
- *Does the network suffer from vanishing gradients for deeper architectures and higher order differential operators?*
  - *Could this be mitigated by using different activation functions?*
- *Is LSE an appropriate loss function?*
- *Why are these methods seemingly so robust to noise in the data?*
- *Can we improve on initializing the network weights or normalizing the data?*
- *How can we quantify the uncertainty associated with our predictions?*

What do the references say? - **Spyros (2021)**

- Time-domain analysis:
  - PINN for modeling the rotor angle for an SMIB system found to be 100 times faster (??)
    - Maybe in some cases PINNs can act faster?

What do the references say? - **Spyros (2022)**

- ACOPF
- PINNs are between 2 to 3 times slower than NNs
- Improvement in performance underwhelming.

Table 2
Neural network properties.

Test case	N_k in hidden layer			Training time	
	G	V	L_m	NN (min)	PINN (xNN)
Case 14	5	10	20	3	2
Case 39	20	30	50	8	2.5
Case 118				88	3.2
Case 162				105	2.6

Table 3
Improving average performance.

Test case		MAE_T (%)	v_g^{avg} (%)	v_{opt}^{avg} (%)	v_{dist}^{avg} (%)
Case 14	NN	0.59	0.02	0.08	5.63
	$PINN_{avg}$	0.42	0.00	0.05	4.43
Case 39	NN	0.89	0.93	0.21	10.78
	$PINN_{avg}$	0.70	0.88	0.08	10.38
Case 118	NN	1.07	1.76	0.31	11.46
	$PINN_{avg}$	0.84	0.76	0.26	10.71
Case 162	NN	2.22	1.42	0.27	37.81
	$PINN_{avg}$	2.00	1.84	0.26	36.72

At a Glance

- In specific problems statements, PINNs found to be a helpful form of Supervised Learning for problem solving.
- PINNs are MORE sample efficient (this makes sense)
- PINNs are overall better for predicting data.
- PINNs are still black boxes.
- PINNs are NOT faster than NNs.
- PINNs do NOT take less training epochs than NNs.
- System equations are NOT restricted to be Partial Differential Equations.
- Reinforcement Learning was NOT covered in this presentation. All references covered here only dealt in Supervised Learning.

Things to do/understand

Feel free to add your inputs here

- Read about other usages of PINNs apart from appending the loss functions with *system equations*
- How do some authors establish formal guarantees on the output of an NN?
- Need to investigate and appreciate *auto-differentiation*.
- I’m yet to see myself how PINNs compare to NNs.

References

- B. Huang and J. Wang, "Applications of Physics-Informed Neural Networks in Power Systems - A Review," in IEEE Transactions on Power Systems, vol. 38, no. 1, pp. 572-588, Jan. 2023, doi: 10.1109/TPWRS.2022.3162473. [link]
- A Hands-on Introduction to Physics-informed Machine Learning [YouTube link]
- Helpful webpage explaining generalizability
- 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. J. Comput. Phys., 378, 686–707. doi: 10.1016/j.jcp.2018.10.045 [link]
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), 359–366. doi: 10.1016/0893-6080(89)90020-8 [link]