Physics Informed Neural Networks (PINN)

Agney K Rajeev
CS460 : Machine Learning

History and Purpose

Since the advent of computers, physics-based simulation models have been widely used to describe, analyze, and predict the behavior of physical systems. Traditionally, these models were derived from the fundamental principles and equations governing the dynamics of these systems, such as Newton's laws of motion, Maxwell's equations, Navier-Stokes equations etc.. In the 2010s, data driven models such as neural networks gained popularity due to their remarkable ability to learn complex patterns and relationships from data. One of the pioneering works in data driven physics modelling was by **M. Raissi**, **P. Perdikaris** and **G.E. Karniadakis** in 2018 detailed in the paper **Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations** [1].
Use of neural networks for physics modelling has its notable shortcomings

- **Ineffectiveness in respecting the laws of physics** : Standard neural networks are almost entirely data-driven and hence might not abide by the fundamental equations of the physical system
- **Unevenly distributed sample data points** : The observed data obtained from the system might not be evenly distributed or just simply lacking in quantity.
- **Inability to extrapolate predictions** : Neural networks are designed to approximate functions only within the region containing the sample points

PINNs were able to mitigate these shortcomings to a large extent using a semi-supervised approach by combining the pattern recognition ability of neural networks along with the differential equations governing the dynamics of a system. They were successful in solving various complex PDEs such as the Burger's equation, Diffusion equation etc. and have been used in the fields of fluid dynamics, quantum mechanics, geophysics and more.

The Algorithm

The **cost function** is a function which attempts to quantify the difference between the approximated function and the true function. Typical regression problems employ the **Mean Squared Error (MSE)** cost function defined as below.

$$L_{NN} = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{1}$$

where f is the neural network function, x_i (**sample points**) are the inputs to f and y_i are the true values at each x_i. The objective of training a Neural Network is to minimize the value of the cost function
PINNs introduce an additional **Differential Cost** term to the cost function. The law of physics which the physical system must adhere to can be expressed as a differential equation of the form Df=0 where D if a differential operator. If so, the differential cost term is given by as

$$L_{Diff} = \frac{1}{m}\sum_{j=1}^{m}((Df)(x_j) - 0)^2 \tag{2}$$

where x_j (called **collocation points**) are sampled from the domain. The values of the differential expression at a collocation point (Df(x_j)) can be calculated by the method of **Automatic Differentiation (Autograd)**.
The overall cost is simply a weighted sum of these two terms.

$$L = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 + k\frac{1}{m}\sum_{j=1}^{m}((Df)(x_j) - 0)^2 \tag{3}$$

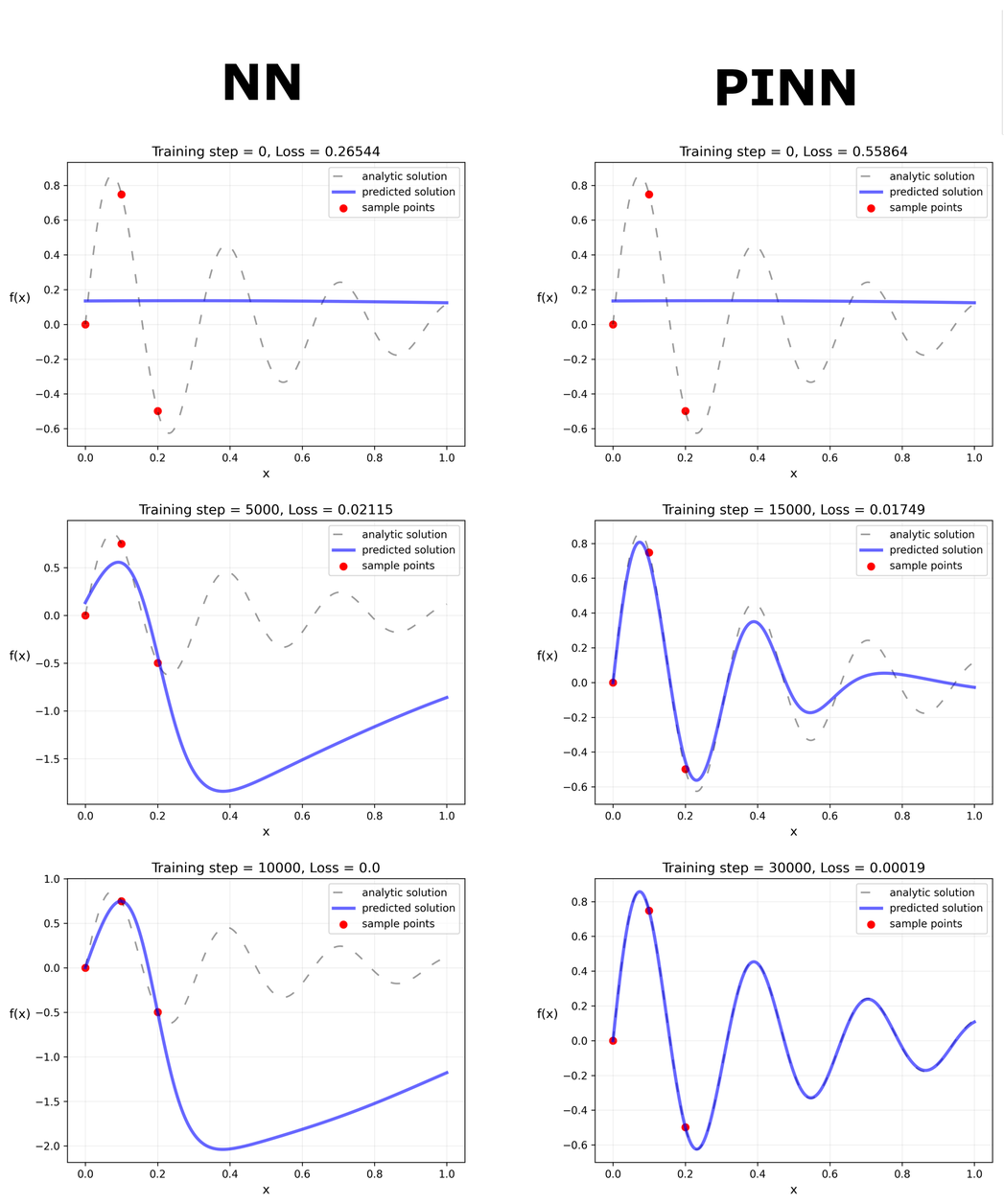
Thus, if the cost function is at a minimum, then the prediction of the neural network will be close to satisfying the differential equation along with the sample data and hence, adhere to the laws of physics. Moreover, the true outputs of the collocation points (y_j) are not present in the cost function. This implies any number of collocation points can be sampled using any distribution from anywhere in the domain.

Sample Code

```python
from torch import zeros_like,ones_like
from torch.autograd import grad

#The derivative calculator for nth degree
def dnfdxn(number_of_derivatives,model,x_values):
    out = [model(x_values)]

    for i in range(number_of_derivatives):
        d = grad(out[-1], x_values, ones_like(out[-1]), create_graph=True)[0]
        out.append(d.view(-1,1))

    return out

#the differential equation is f''(x) + 2Gf'(x) + R^2f(x) = 0 where G and R are constants

#Creates the cost function using the necessary derivatives
def cost_func_maker(f,x_data,y_data):
    # f is the neural network model
    # x_data is the tensor containing the sample points
    # y_data is the tensor containig the function values at the sample points

    #The cost function is a sum of MSE Loss due to the sample points and the differential equation constraint
    def cost_func(x):

        x0 = x_data
        f0 = y_data
        samplecost = f(x0) - f0 #cost due to the sample points

        f_value, dfdx, d2fdx2 = dnfdxn(2,f,x)
        DEcost = d2fdx2 + 2*G*dfdx + (R**2)*f_value   #cost due to differential equation constraint

        cost = nn.MSELoss()
        cost_val =  cost(samplecost,zeros_like(samplecost)) + (1e-4)*cost(DEcost,zeros_like(DEcost))

    return cost_func
```

Example : Damped Simple Harmonic Oscillator

$$f''(x) + 2Gf'(x) + R^2 f(x) = 0 \tag{4}$$



Fields of Application

- Fluid Dynamics [2]
- Aerodynamics [3]
- Weather Prediction [4]
- Medical Imaging [5]
- Combustion Science [6]
- Signal Processing [7]

References

1. Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics (Print), 378, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
2. Омарова, П., Amirgaliyev, Y., Kozbakova, A., & Ataniyazova, A. (2023). Application of Physics-Informed Neural Networks to river silting Simulation. Applied Sciences, 13(21), 11983. https://doi.org/10.3390/app132111983
3. Ang, E. H., & Ng, B. F. (2022). Physics-Informed neural networks for flow around airfoil. AIAA SCITECH 2022 Forum. https://doi.org/10.2514/6.2022-0187
4. Kashinath, K., Mustafa, M. E., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., Manepalli, A., Chirila, D. B., Yu, R., Walters, R., White, B., Xiao, H., Tchelepi, H. A., Marcus, P., Anandkumar, A., . . . Prabhat. (2021). Physics-informed machine learning: case studies for weather and climate modelling. Philosophical Transactions of the Royal Society A, 379(2194), 20200093. https://doi.org/10.1098/rsta.2020.0093
5. Van Herten, R. L. M., Chiribiri, A., Breeuwer, M., Veta, M., & Scannell, C. M. (2022). Physics-informed neural networks for myocardial perfusion MRI quantification. Medical Image Analysis, 78, 102399. https://doi.org/10.1016/j.media.2022.102399
6. Hosseini, V. R., Mehrizi, A. A., Güngör, A., & Afrouzi, H. H. (2023). Application of a physics-informed neural network to solve the steady-state Bratu equation arising from solid biofuel combustion theory. Fuel, 332, 125908.https://doi.org/10.1016/j.fuel.2022.125908
7. Russell, M., & Wang, P. (2022b). Physics-informed deep learning for signal compression and reconstruction of big data in industrial condition monitoring. Mechanical Systems and Signal Processing, 168, 108709. https://doi.org/10.1016/j.ymssp.2021.108709