MACHINE LEARNING IN PHYSICS
PHYSICS-INFORMED NEURAL NETWORKS (PINN)

HARRISON B. PROSPER

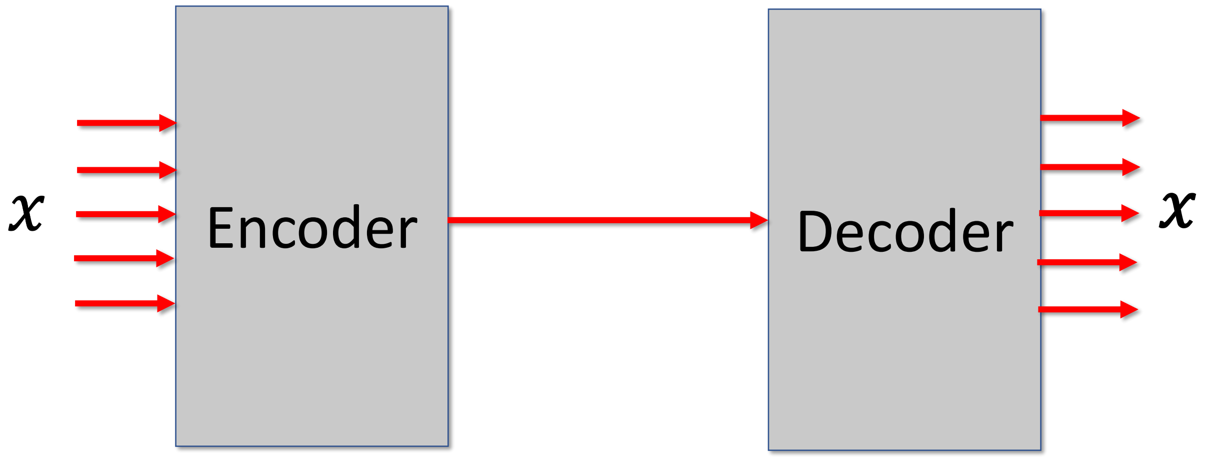
PHY6938

Recap: Autoencoder

An autoencoder *compresses* the input data x to a latent space z and then back to x.

$h(x) \to z$ $g(z) \to x$

$z \in \mathbb{R}^n$



Introduction

This week we introduce a neural network called a:

1. Convolutional neural networks (CNN)
2. Autoencoder (AE)
3. Physics-informed neural networks (PINN)
4. Flow and diffusion models
5. Graph neural networks (GNN)
6. Transformer neural networks (TNN)

Introduction

- The method of physics-informed neural networks (PINN) is a way[1,2] to solve ordinary and partial differential equations (ODE, PDE) using neural networks.

- The method leverages the modeling flexibility of neural networks and the computational tools that have been developed to fit these enormously complicated functions.

1. Raissi, M., Perdikaris, P., Karniadakis, G.E., Physics-informed neural networks: A deep learning frame-work for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707 (2019).
2. Cuomo, S *et al*. Journal of Scientific Computing (2022) 92:88 https://doi.org/10.1007/s10915-022-01939-z

Introduction

Advantages of the PINN method

- Provides mesh-free solutions.
- Can incorporate data to guide solutions.
- Scales well with the dimensionality of the space.
- Can find solutions conditioned on initial/boundary conditions. In effect, solutions can be found for *infinitely* many initial/boundary conditions at the same time.

Disadvantages of PINNs

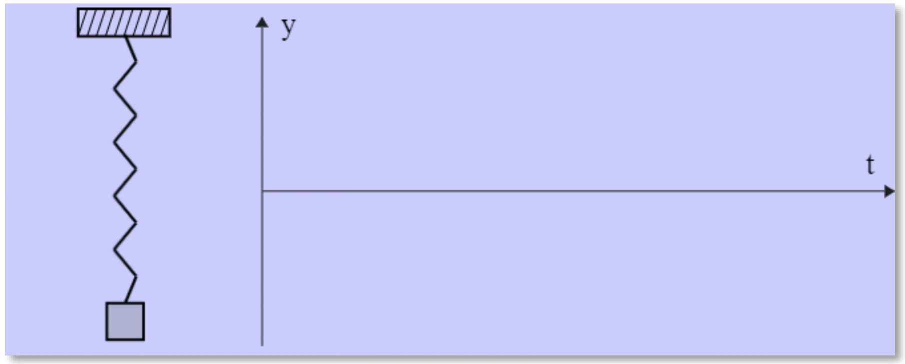
- For a given set of initial/boundary conditions, much slower than traditional numerical methods.
- Convergence is sensitive to the details of the average loss.

Introduction

The essential ideas of PINNs will be explained using the well-known problem of the damped harmonic oscillator,

$$\frac{d^2u}{dt^2} + 2\lambda\frac{du}{dt} + \omega_0^2 u = 0, \qquad \lambda = \frac{\mu}{m}, \omega_0 = \sqrt{\frac{k}{m}}$$

where u is the displacement of an object of mass m, k is the spring constant, μ the coefficient of kinetic friction, and ω_0 the natural frequency of the spring.



https://www.mathwarehouse.com/harmonic-motion/interactive-damped-oscillator.php

Introduction

The dynamics of the damped harmonic oscillator can be described with a single free parameter $\alpha = \lambda/\omega_0$, which allows us to write

$$\frac{d^2u}{dz^2} + 2\alpha\frac{du}{dz} + u = 0$$

where the dimensional time is $z = \omega_0 t.$

Since this is a 2nd order ODE, to get a particular solution we need to specify initial conditions:

$$u_0 = u(0), \qquad v_0 = \left.\frac{du}{dz}\right|_{z=0}$$

Introduction

The damped harmonic oscillator can be solved exactly.

We'll consider the case $\alpha < 1$, for which the solution is

$$u(z) = \frac{u_0}{\cos\phi}\exp(-\alpha z)\cos\left(z\sqrt{1-\alpha^2}+\phi\right)$$

$$\phi = \arctan\left(-\frac{(\alpha+v_0/u_0)}{\sqrt{1-\alpha^2}}\right)$$

- The exact solution can be used to check the accuracy of the neural network solutions.
- We'll compute solutions *conditioned* on the triplet $(\alpha, u_0, v_0) \in S \subset \mathbb{R}^3$

PHYSICS-INFORMED NEURAL NETWORKS (PINN)

PINNs: Basic Idea

1. Model the solution $u(x)$ with a neural network $f(x, \theta)$, where $x = (z, u_0, v_0, \alpha)$ and θ are the network parameters. How?

2. By minimizing an average loss function comprising a *weighted sum* of the three components:
   1. $R_{ODE}(\theta)$ imposes an ODE-based constraint;
   2. $R_C(\theta)$ imposes initial/boundary conditions,
   3. $R_D(\theta)$ imposes constraints provided by data.

PINNs: Basic Idea

The overall average loss function is a weighted sum of

$$R_{ODE}(\theta) = \frac{1}{N_{ODE}} \sum_{i=1}^{N_{ODE}} \left[\mathcal{F}\left(f(x_i, \theta)\right)\right]^2$$

$$R_C(\theta) = \left(f(x_i, \theta)\Big|_{z=0} - u_0\right)^2 + \left(\frac{df(x_i, \theta)}{dz}\Bigg|_{z=0} - v_0\right)^2$$

$$R_D(\theta) = \frac{1}{N_D} \sum_{i=1}^{N_D} \left[f(x_i, \theta) - u(x_i)\right]^2$$

$\mathcal{F}\left(f(x_i, \theta)\right) = 0$ is the ODE, $x = (z, u_0, v_0, \alpha)$.

PINNs: Problem of Weights

One problem with the average loss

$$R(\theta) = w_{ODE}\, R_{ODE}(\theta) + w_C\, R_C\,(\theta) + w_D\, R_D(\theta)$$

is that minimization can be sensitive to the relative weights of the three components.

Moreover, since the different terms are in competition the accuracy of the solution also depends on the choice of weights.

PINNs: Ansatz

One way to reduce the sensitivity to the weights w_{ODE}, w_C, and w_D is to eliminate w_C by incorporating the initial conditions into the solution explicitly.

For example, we could model the solution $u(z)$ using the *ansatz*

$$u(z) = u_0 + v_0\, z + f(x, \theta) z^2$$

For the damped harmonic oscillator problem, we'll use

$$u(z) = \frac{u_0 + v_0\, z + f(x, \theta) z^2}{1 + z^2}$$

PINNs: Automatic Differentiation

The key technology that makes the PINN method feasible is automatic differentiation (autodiff), that is, the technology that is used to compute the gradients, *exactly*, of the average loss with respect to the parameters θ of the neural network.

We can use autodiff to compute the derivatives with respect to z in the average loss functions

$$R_{ODE}(\theta) = \frac{1}{N_{ODE}} \sum_{i=1}^{N_{ODE}} [\mathcal{F}(f(x_i, \theta))]^2$$

$$R_C(\theta) = \left(f(x_i, \theta)\Big|_{z=0} - u_0 \right)^2 + \left(\frac{df(x_i, \theta)}{dz}\Bigg|_{z=0} - v_0 \right)^2$$

PINNs: Autodiff

Compute

$$\nabla_x u = \left(\frac{\partial u}{\partial z}, \frac{\partial u}{\partial u_0}, \frac{\partial u}{\partial v_0}, \frac{\partial u}{\partial \alpha}\right)$$

using

```
dudx = torch.autograd.grad(u, x,
                           torch.ones_like(u),
                           create_graph=True)[0]
```

Pick off *du/dz* and change its shape.

```
dudz = dudx[:,  0].view(-1, 1)
```

PINNs: Automatic Differentiation

Compute

$$\nabla \frac{\partial u}{\partial z} = \left(\frac{\partial^2 u}{\partial z^2}, \frac{\partial^2 u}{\partial u_0 \partial z}, \frac{\partial^2 u}{\partial v_0 \partial z}, \frac{\partial^2 u}{\partial \alpha \partial z} \right)$$

```
d2u = torch.autograd.grad(dudz, x,
                          torch.ones_like(u),
                          create_graph=True)[0]
```

Pick off $\frac{\partial^2 u}{\partial z^2}$ and change its shape.

```
d2udz2 = d2u[:, 0].view(-1, 1)
```

PINNs: Note on implementation

- In the implementation just sketched, we are computing many more derivatives than are needed!

- However, by feeding the dimensionless time z and u_0, v_0, α into *separate* neural networks, whose outputs are then combined, it is possible to compute only the derivatives that are relevant.

PINNs: Average Loss

Compute average loss $R(\theta)$.

First pick off *alpha*, change its shape,

```
alpha = x[:, -1].view(-1, 1)
```

then compute $O = \frac{d^2u}{dz^2} + 2\alpha\frac{du}{dz} + u$

```
O = d2udz2 + 2*alpha*dudz + u
R = torch.mean(O**2)
```

PINNs: Model / Training

Model

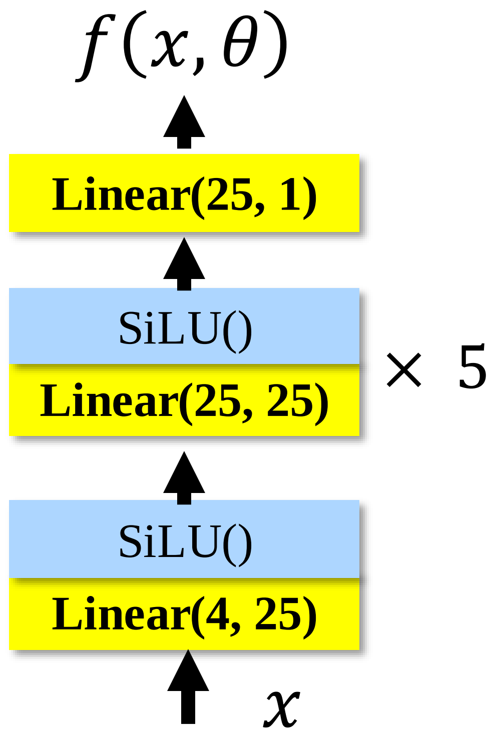
$f(x,\theta),\ x = (z, u_0, v_0, \alpha).$

Domain

$x \in [0, 20] \otimes [-1, 1] \otimes [-2, 2] \otimes [0, 0.5]$

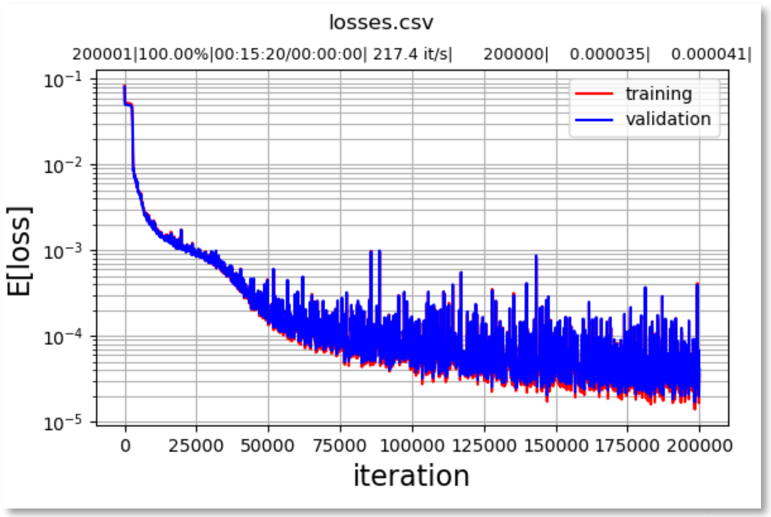
Training

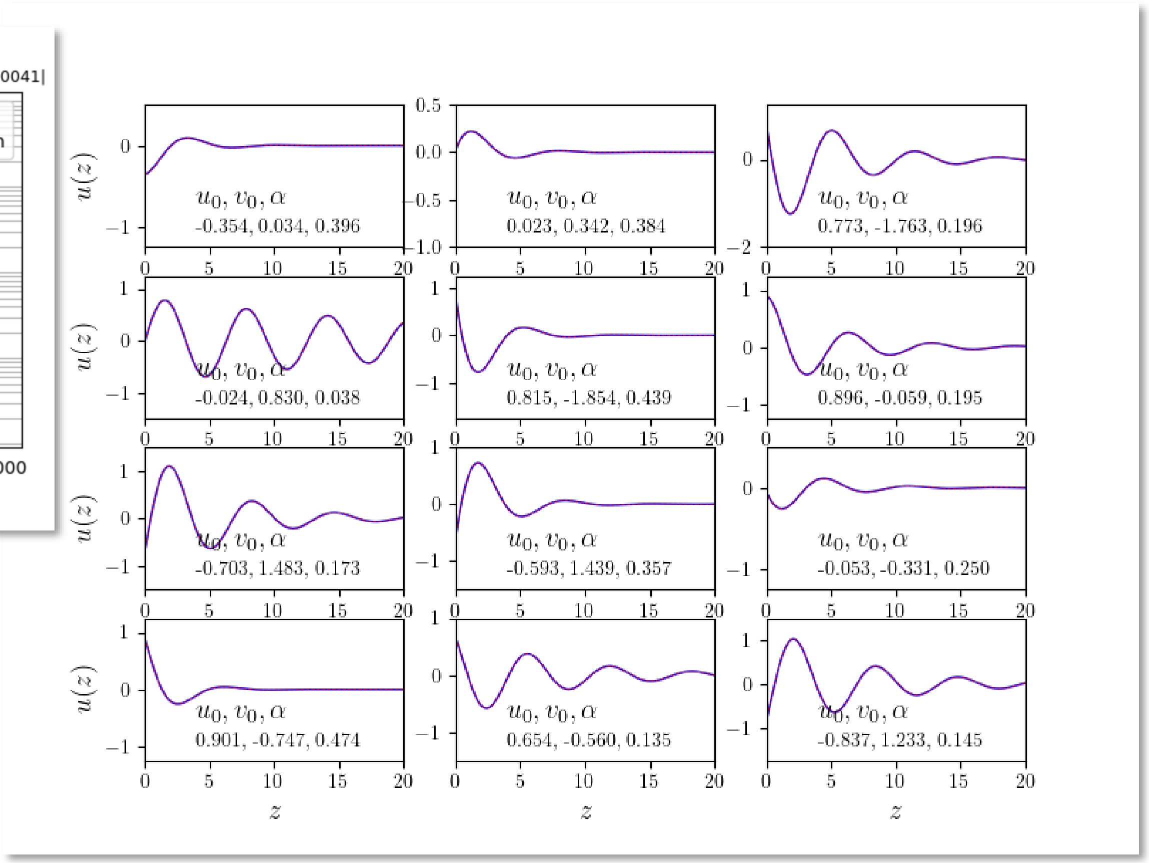
1. Sample size: 60,000
2. Learning rate: 10^{-3}
3. Batch size: 256
4. Iterations: 200,000



PINNs: Results

Comparisons of PINN results with exact solution for a random collection of points (u_0, v_0, α).





Summary

- So-called physics-informed neural networks offer a new way to solve differential equations in many scientific and engineering domains.
- Finding solutions with PINNs is typically more computationally demanding than with traditional numerical methods.
- However, an infinite number of solutions can be found at the same time, each corresponding to a different set of initial/boundary conditions and problem parameters.