Missouri University of Science and Technology

A Hands-on Introduction to Physics-Informed Neural Networks

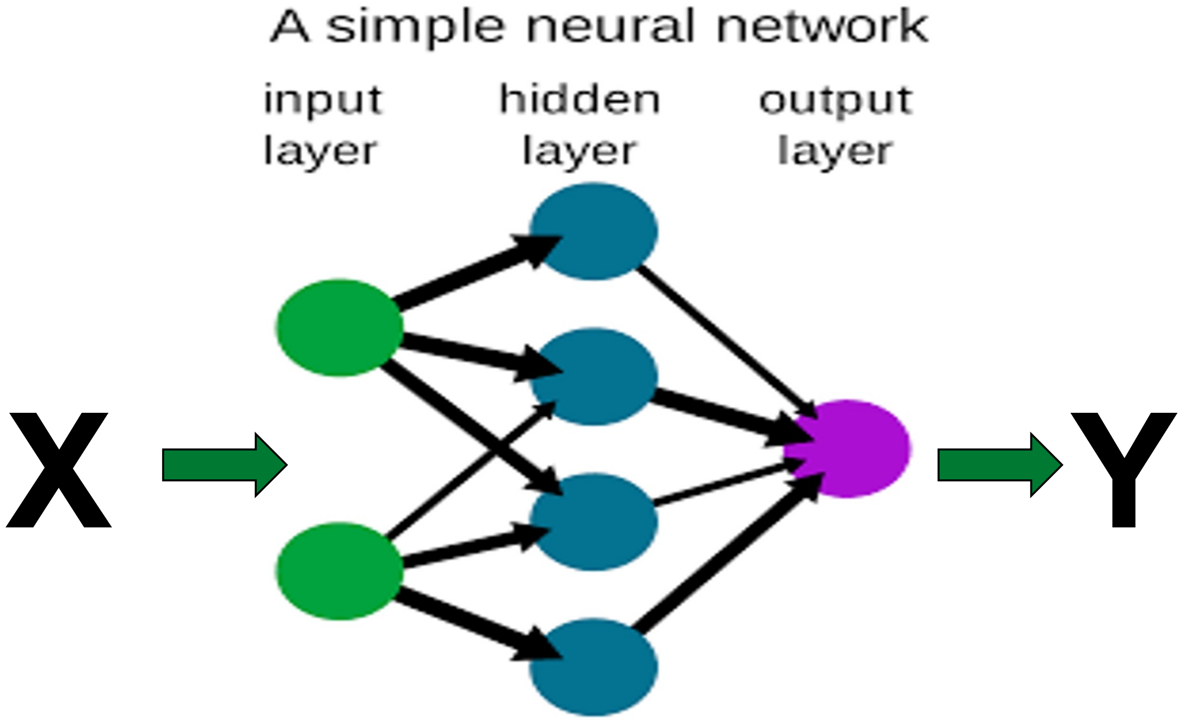
Ronit Das

May 20, 2024

Objective

How can we incorporate physical information in the form of differential equations to regularize neural networks?

Reminder: what's a neural network?



Reminder: how to train a NN?

1. Mainly trained using **maximum likelihood estimation**
2. **MLE** takes the form of *squared L2 loss* when the input, **X** ~ N($\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$) (*regression*)
3. **MLE** is equivalent to *crossentropy loss* (used mainly in classification)
4. NN parameters are updated using an optimization algorithm:
   a. **Gradient Descent (stochastic)** :

Reminder: how to train a NN?

$$\text{Repeat Until Convergence } \{$$

$$\omega \leftarrow \omega - \alpha * \nabla_w \sum_{1}^{m} L_m(w)$$

$$\}$$

batch size

learning rate

performed using Automatic
Differentiation in Tensorflow, Pytorch

Illustrative Example: Solving an ODE

- Is solving a Differential Equation using Artificial Neural Network completely new?
    - **NO!**

Lagaris, Isaac E., Aristidis Likas, and Dimitrios I. Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations." *IEEE transactions on neural networks* 9.5 (1998): 987-1000.

From ODE to loss function

- Consider the IVP:

$$\frac{d\Psi}{dx} = f(x, \Psi) \qquad \Psi(0) = V$$

- Can analytically satisfy the initial condition, by parameterizing $\boldsymbol{\theta}$

$$\hat{\Psi}(x;\theta) = V + x \cdot DNN(x;\theta)$$

From ODE to loss function

- But how do we learn $\boldsymbol{\theta}$?
  - **minimize** the *Integrated Squared Residual* of the ODE:
    - $$J(\theta) = \int_0^1 \left[\frac{d}{dx}\left(\hat{\Psi}(x;\theta)\right) - f\left(x,\ \hat{\Psi}(x;\theta)\right)\right]^2 dx$$
    - weight update rule (Robbins, Monro (1951)):

$$\theta_{t+1} = \theta_t - \frac{\alpha_t}{m}\sum_{j=1}^{m} \nabla_\theta \left[\frac{\mathrm{d}}{\mathrm{d}x}\left(\hat{\Psi}(x_t;\theta_t)\right) - f\left(x_t,\ \hat{\Psi}(x_t;\theta_t)\right)\right]^2$$

MISSOURI S&T

Illustrative Example: Solving a PDE

- Elliptic PDE :
  - $$-\nabla[a(x)\nabla u(x)] + c(x)u(x) = f(x)$$
  - parameterize $u(x, ; \theta)$ with a NN to satisfy, say the Dirichlet Boundary Condition
  - **Solve?**
    - *Squared Integral Approach*:

$$\min. \int \left[\nabla[a(x)\nabla \hat{u}(x;\theta)] + c(x)\hat{u}(x;\theta) + f(x)\right]^2 dx$$

From PDE to loss function

- the *Squared Integral Approach* does not necessarily give us a *unique solution*
- Need something better!
  - **Dirichlet's Principle:**
    - $$\min. \int \left[\frac{1}{2} a(x) \nabla \hat{u}(x;\theta) + c(x)\hat{u}^2(x;\theta) - f(x)\hat{u}(x;\theta) \right] dx - \int_{\Gamma_S} g_S \hat{u}(x;\theta) d\Gamma_S$$
    - also called the *energy equation*
      - gives a *unique solution* to satisfy the PDE (same thing also holds for ODE)

Motivation for using DNN to solve ODE/PDEs

- A single ODE/PDE can be solved using numerical approximation methods. Then why look into NNs?
  - You can solve for multiple number of parameters, with multiple different boundary conditions, **in a single sweep!**

Some of its applications

- high-dimensional uncertainty propagation through PDEs
- solve problems having *free boundaries* and also *Stefan problem*
- inverse / model calibration problems
- data filtering
- ….

Will it work everytime?

- Short answer:
  - No!
  - **Problems:**
    - *vanishing gradients*
    - *spectral bias*
      - DNN first finds the low-frequency components
      - the high-frequency parts take forever to learn
    - ……

Practical Applications

Time for some hands-on training for PINNs…………

Any questions????