Recovering Governing Equations of PDEs via Friedrichs Learning

Jonathn Chang [1] Mentors: Dr. Chunmei Wang [2] and Dr. Haizhao Yang [3]

[1]University of California, Los Angeles [2]University of Florida [3]University of Maryland

This material is based upon work supported by the National Science Foundation under Grant No. DMS-2050133.

Motivation

Data-driven recovery of ODE/PDEs has been a field of interest in recent years. However, most existing learning algorithms focus on the strong form of equations, which involves derivatives of observations that may be noisy, discontinuous, or collected non-uniformly. These algorithms typically encounter inaccuracies due to the inability to capture the derivative of discontinuous data, the highly ill-conditioned evaluation of derivatives of noisy data, and instability errors in approximating derivatives of non-uniform data. Here, we propose Friedrichs Learning to solve these challenges, which reformulates the problem of recovering time-dependent PDEs into a minimax optimization problem that uses deep learning and the weak form of the PDE.

Minimax Formulation

The minimax formulation used in Friedrichs Learning will be introduced with the time-dependent reaction diffusion PDE in one dimension,

$$\frac{\partial u}{\partial t} = \frac{d}{dx}\left(H\frac{\partial u}{\partial x}\right) + Gu. \tag{1}$$

The goal is to recover H and G as functions of x given data for u, where u is defined on a bounded domain $\Omega \times T$ with boundary $\partial\Omega$ and ∂T. Friedrichs Learning uses the weak form of the equation which involves a smooth test function $v \in \mathcal{H}_0^1(\Omega) \times \mathcal{H}_0^1(T)$. We denote the inner product on Ω and T as

$$(u, v)_{\Omega,T} = \int_T \int_\Omega uv \, dxdt \tag{2}$$

To convert (1) into its weak form, we take the inner product of both sides with respect to the test function v and integrate by parts the terms that involve derivatives in u,

$$\int_T \left[\left[uH\frac{\partial v}{\partial x}\right]_{\partial\Omega} - \int_\Omega u\frac{\partial v}{\partial t}\,dx\right] dt = \int_T \int_\Omega \left[u\frac{d}{dx}\left(H\frac{\partial v}{\partial x}\right) + Guv\right] dxdt. \tag{3}$$

We use this to formulate the supervised learning problem into a minimax optimization problem with v, H, and G parameterized as DNNs $v^{\theta_v} = v(x,t;\theta_v)$, $H^{\theta_H} = H(x;\theta_H)$, and $G^{\theta_G} = G(x;\theta_G)$,

$$(\bar{\theta}_H, \bar{\theta}_G, \bar{\theta}_v) = \arg\min_{\theta_H,\theta_G}\max_{\theta_v} \mathscr{L}(\theta) =$$
$$= \arg\min_{\theta_H,\theta_G}\max_{\theta_v} \frac{1}{||v^{\theta_v}||_{L^2}} \left| \int_T \left[\left[uH^{\theta_H}\frac{\partial v^{\theta_v}}{\partial x}\right]_{\partial\Omega} - \int_\Omega \left[u\frac{\partial v^{\theta_v}}{\partial t} + u\frac{d}{dx}\left(H^{\theta_H}\frac{\partial v^{\theta_v}}{\partial x}\right) + G^{\theta_G}uv^{\theta_v}\right] dx\right] dt \right|, \tag{4}$$

where $(\bar{\theta}_H, \bar{\theta}_G, \bar{\theta}_v)$ is the set of parameters that solves the minimax problem. The normalization term $||v^{\theta_v}||_{L^2}$ guarantees that the solution of the minimax problem is well-posed.

Neural Network Structure

To solve the minimax problem in (4), we employ a ResNet structure to parameterize each of the DNNs. This is defined recursively as follows.

$$\begin{aligned} \boldsymbol{h}_0 &= \boldsymbol{V}\boldsymbol{x} \\ \boldsymbol{g}_l &= \sigma(\boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{b}_l) \\ \boldsymbol{h}_l &= \boldsymbol{g}_l + \boldsymbol{h}_{l-1} \\ \phi(\boldsymbol{x};\theta) &:= \boldsymbol{a}^T\boldsymbol{h}_L \end{aligned} \tag{5}$$

where $l = 1, \ldots, L$, and

- $\boldsymbol{V} \in \mathbb{R}^{m\times d}$ (input layer)
- $\boldsymbol{W}_l \in \mathbb{R}^{m\times m}$ (lth weight matrix)
- $\boldsymbol{b}_l \in \mathbb{R}^m$ (lth bias vector)
- $\boldsymbol{a} \in \mathbb{R}^{m\times 1}$ (output layer)
- σ is either Tanh or $\text{ReLU}^k := \max\{0, x\}^k$ (k depends on the differentiability requirement of the function being parameterized)
- d is the input dimension of the function being parameterized
- m and L are the width and hidden layer depth of the DNN, respectively

For the boundary requirement of the DNN for $v(x,t;\theta_v)$, suppose $\hat{\phi}(x,t;\theta)$ is the output described in (5). Then, we can define

$$v(x,t;\theta_v) = a(x,t)\hat{\phi}(x,t;\theta) \tag{6}$$

where $a(x,t)$ is a function specifically constructed such that $a(x,t) = 0$ on $\partial\Omega$ and ∂T.

Network Training

The networks are trained to solve the minimax problem in (4) using gradient descent (Adam). In the training algorithm, there are n outer iterations and n_t and n_s inner iterations for updating θ_v and (θ_H, θ_G), respectively. In each inner iteration n_t, we hold the parameters (θ_H, θ_G) constant while updating the parameters θ_v according to $\theta_v \leftarrow \theta_v + \eta_v g_t$, where η_v is the step size associated with v at the specified outer iteration and g_t is the gradient of the loss function $\mathscr{L}(\theta)$ given in (4), calculated with respect to θ_v. This gradient is evaluated via autograd in Pytorch.

Similarly, in each inner iteration n_s, we hold the parameters θ_v constant while updating the parameters θ_H according to $\theta_H \leftarrow \theta_H - \eta_H g_s$ and θ_G according to $\theta_G \leftarrow \theta_G - \eta_G g_s$, where g_s is calculated with respect to (θ_H, θ_G). The learning rates for each network decrease after every outer iteration according to the following exponential decay scheme,

$$\eta_v^{(k)} = \eta_v^{(0)}\left(\frac{1}{10}\right)^{k/v_v}, \tag{7}$$

where v_v is the decay rate for θ_v. This follows identically for η_H and η_G. The table of all the hyperparameters required for training are listed below.

Parameter	Definition	Parameter	Definition
n	number of outer iterations	L	depth of each ResNet network
n_t	number of inner iterations for v	m_t	width of each layer in v
n_s	number of inner iterations for H and G	m_s	width of each layer in H and G
$\eta_v^{(0)}$	initial learning rate for v	v_v	decay rate for η_v
$\eta_H^{(0)}$	initial learning rate for H	v_H	decay rate for η_H
$\eta_G^{(0)}$	initial learning rate for G	v_G	decay rate for η_G

Table: Hyperparameters for Training

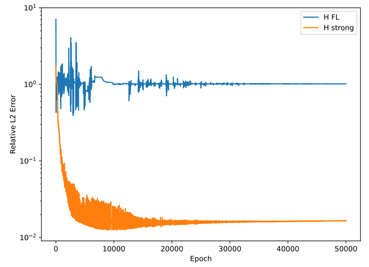
Friedrichs Learning Tests

Let $\Omega = [0,1]$ and $T = [0,2]$, with the boundary and initial value conditions $u(0,t) = 0$, $u(1,t) = 0$, and $u(x,0) = x(1-x)$, and the following choices of H and G.

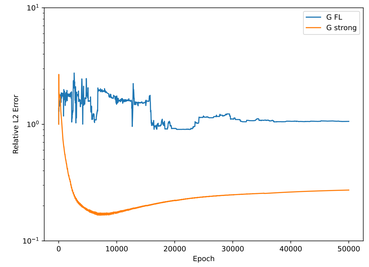
$$\begin{aligned} H(x) &= \frac{e^{-\sin(2\pi x)}}{20} \\ G(x) &= \cos(2\pi x) \end{aligned} \tag{8}$$

For the choice of $a(x,t)$ in (6), we choose $a(x,t) = \frac{x(1-x)t(2-t)}{0.5^2}$, which is 0 on the spatial and temporal boundaries.

The relative L^2 error produced by Friedrichs Learning is two orders of magnitude worse in H and one order of magnitude worse in G compared to the error produced by a mean-squared error loss function in the strong form.



(a) Error History for H



(b) Error History for G

Other Tests

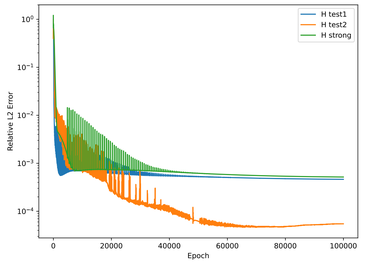
Through extensive testing, we discover that a choice of $v \notin \mathcal{H}_0^1\Omega \times \mathcal{H}_0^1(T)$ may be more favorable, and we construct two new loss functions to further test the weak form. The first uses a similar mean-squared scheme to the strong form, but multiplies the test function v to each term:

$$(\bar{\theta}_H, \bar{\theta}_G, \bar{\theta}_v) = \arg\min_{\theta_H,\theta_G}\max_{\theta_v} \mathscr{L}(\theta) =$$
$$= \arg\min_{\theta_H,\theta_G}\max_{\theta_v} \mathbb{E}_{(x,t)\in\Omega\times T}\left[\left|\left(\frac{\partial u}{\partial t} - \frac{d}{dx}\left(H(x;\theta_H)\frac{\partial u}{\partial x}\right) - G(x;\theta_G)u\right)v(x,t;\theta_v)\right|^2\right]. \tag{9}$$

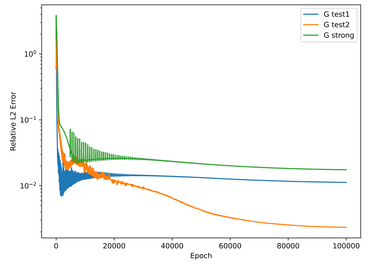
The second proceeds with the inner product in the weak form, but does not perform any integration by parts and places the absolute value on the inside:

$$(\bar{\theta}_H, \bar{\theta}_G, \bar{\theta}_v) = \arg\min_{\theta_H,\theta_G}\max_{\theta_v} \mathscr{L}(\theta) =$$
$$= \arg\min_{\theta_H,\theta_G}\max_{\theta_v} \frac{1}{||v^{\theta_v}||_{L^2}} \int_T \int_\Omega \left|\left(\frac{\partial u}{\partial t} - \frac{d}{dx}\left(H(x;\theta_H)\frac{\partial u}{\partial x}\right) - G(x;\theta_G)u\right)v(x,t;\theta_v)\right| dxdt. \tag{10}$$

These functions are tested again against the strong form loss function in an experiment for which the exact solution is known and its derivatives are smooth. The two new loss functions both outperform the strong form, with (10) improving on the strong form by around one order of magnitude. Also, the graphs of both new loss functions are much more consistent in converging to a minimum compared to the previous behavior of the graphs of FL.



(a) Error History for H



(b) Error History for G

Despite the unfavorable results with the weak form in Friedrichs Learning, we were able to achieve far better results with these new loss functions, indicating merit in utilizing a test function v to reformulate the recovery problem into a minimax optimization problem. From here, further research will be conducted to explore the new loss functions and formulate a new weak form without boundary conditions on v that can be tested against noisy and non-uniform solution data.