Implicit Form Neural Network for Learning Scalar Hyperbolic Conservation Laws

Xiaoping Zhang $^1$, Tao Cheng $^1$ and Lili Ju $^2$

$^1$School of Mathematics and Statistics, Wuhan University
$^2$Department of Mathematics, University of South Carolina

Mathematical and Scientific Machine Learning, 2021
August 19, 2021
In recent years, deep learning techniques have been applied to mathematical and scientific computing problems, and a large number of excellent research works on learning solutions of PDEs have emerged.

- The first attempt of solving PDEs using NNs can be traced back at least to the late last century. ¹. However, due to the limitation of computational resource back that time, it unfortunately did not attract much attention of researchers.

- Currently, there are two major ways to use deep learning for numerical solutions of PDEs:

  1. **Deep Ritz method** ² and its variants Convert the PDEs into their equivalent variational forms and then solve them using NNs

  2. **PINN** ³ and its variants Use NNs to directly deal with the original PDEs

---

Introduction

Scalar hyperbolic conservation law

Let \( \mathbf{x} = (x_1, \cdots, x_d)^T \in \mathbb{R}^d \) and \( \mathbf{f} = (f_1, \cdots, f_d)^T \in \mathbb{R}^d \), the scalar hyperbolic conservation law can be formulated as follows:

\[
\begin{cases}
    u_t + \nabla \cdot \mathbf{f}(u) = 0, & (\mathbf{x}, t) \in \mathbb{R}^d \times (0, T], \\
    u(\mathbf{x}, 0) = \phi(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^d.
\end{cases}
\]  

(1a)  

(1b)

where \( u \) is an unknown function defined in \( \mathbb{R}^d \).

Remark

Although the deep Ritz method and PINN have achieved great success, they still exhibit poor performance in handling problems with strong discontinuous solutions, such as hyperbolic conservation laws.
Figure: Reference solution and predicted solution of PINN for Burgers’ equation $u_t + uu_x - \epsilon u_{xx} = 0$ when $t = 0.5$
Theorem

An implicit form for the solution of (1) can be formulated as

$$u = \phi(x - f'(u)t),$$

(2)

where $f'$ denotes the velocity

$$f'(u) = (f'_1(u), \cdots, f'_d(u))^T.$$  

(3)

Contribution

A fully-connected neural network 'IFNN' is proposed to learn the solution of (1) by using the loss function based on the implicit form (2) instead of the original equation (1).
Network model with parameters \( \Theta = \{ W^k, b^k \}_{k=1}^D \).

\[
u(x, t; \Theta) = (\ell_D \circ \sigma \circ \ell_{D-1} \cdots \circ \sigma \circ \ell_1)(x, t), \quad \ell_k(v^{k-1}) = W^k v^{k-1} + b^k. \quad (4)\]

Residual block

\[
v^k = v^{k-1} + \sigma \circ \ell_k(v^{k-1})\]
Sampling strategy

To train the network \((4)\), we need a set of sampling points with respect to \((x, t)\), and the Latin Hypercubic Sampling (LHS) is used to generate the point set \(\mathcal{D}\), and then split it into three parts:

- \(\mathcal{D}_{IC} = \mathcal{D} \cap (\Omega \times \{0\})\),
- \(\mathcal{D}_{BC} = \mathcal{D} \cap (\partial\Omega \times (0, T])\),
- \(\mathcal{D}_{IM} = \mathcal{D} \setminus (\mathcal{D}_{IC} \cup \mathcal{D}_{BC})\).
Loss function

\[ \mathcal{L} (\Theta) = \mathcal{L}_{IM} (\Theta) + \lambda_1 \mathcal{L}_{IC} (\Theta) + \lambda_2 \mathcal{L}_{BC} (\Theta), \]

where

\[ \mathcal{L}_{IM} (\Theta) = \frac{1}{|D_{IM}|} \sum_{(x,t) \in D_{IM}} \left[ u(x, t; \Theta) - \phi(x - f'(u)t) \right]^2, \]

\[ \mathcal{L}_{IC} (\Theta) = \frac{1}{|D_{IC}|} \sum_{(x,t) \in D_{IC}} \left[ u(x, t; \Theta) - \phi(x) \right]^2. \]

\[ \mathcal{L}_{BC} (\Theta) = \frac{1}{|D_{BC}|} \begin{cases} 
\sum_{(x,t) \in D_{BC}} \left[ u(x, t; \Theta) - g_d(x, t) \right]^2, & \text{Dirchlet B.C.} \\
\sum_{(x,t) \in D_{BC}} \left[ \frac{\partial u}{\partial n} (x, t; \Theta) - g_n(x, t) \right]^2, & \text{Neumann B.C.} \\
\sum_{(x,t) \in D_{BC}} \left[ u(x, t; \Theta) - u(x', t; \Theta) \right]^2, & \text{Periodic B.C.} 
\end{cases} \]

where \( x' \) is the symmetric point of \( x \) in the opposite boundary side of \( \Omega \).
Numerical Experiments

Implementation Detail

- Our IFNN is implemented using PyTorch and we use 6 hidden layers with 20 neurons per each hidden layer for all tests.
- Adam optimizer is used to train the network and the number of epochs is set as 25000. The learning rate is initially set to 0.001, and then adjusted with a decay rate of 0.7 per 3000 epochs.
- The hyperparameters $\lambda_1$ and $\lambda_2$ in the loss function are both chosen as 1.
Numerical Experiments - 1D inviscid Burgers’ equation with shock wave (I)

IC: \( \phi(x) = \begin{cases} 1, & x \leq 0 \\ 0, & x > 0 \end{cases} \), Dirichlet BC: \( u(-1, t) = 1 \)

<table>
<thead>
<tr>
<th>Model</th>
<th>Activation</th>
<th>( \text{tanh} )</th>
<th>( \text{sin} )</th>
<th>( \text{ReLU} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>PINN</td>
<td></td>
<td>( 1.17 \times 10^{-1} )</td>
<td>( 1.54 \times 10^{-1} )</td>
<td>( 1.05 \times 10^{0} )</td>
</tr>
<tr>
<td>IFNN</td>
<td></td>
<td>( 2.53 \times 10^{-4} )</td>
<td>( 6.63 \times 10^{-2} )</td>
<td>( 8.01 \times 10^{-2} )</td>
</tr>
</tbody>
</table>

**Table:** \( L_{2} \) errors of IFNN with different activation functions

<table>
<thead>
<tr>
<th>( #l )</th>
<th>( #n )</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>( 5.76 \times 10^{-2} )</td>
<td>( 2.33 \times 10^{-2} )</td>
<td>( 9.23 \times 10^{-3} )</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>( 8.91 \times 10^{-3} )</td>
<td>( 2.72 \times 10^{-3} )</td>
<td>( 6.71 \times 10^{-4} )</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>( 2.97 \times 10^{-3} )</td>
<td>( 2.50 \times 10^{-4} )</td>
<td>( 2.32 \times 10^{-4} )</td>
</tr>
</tbody>
</table>

**Table:** \( L_{2} \) errors of IFNN with different number of hidden layers (\( \#l \)) and neurons (\( \#n \)).

**Figure:** The effect of the size of training set on the prediction accuracy of IFNN.
Numerical Experiments - 1D inviscid Burgers’ equation with shock wave (I)

IC: $\phi(x) = \begin{cases} 
1, & x \leq 0 \\
0, & x > 0 
\end{cases}$, Dirichlet BC: $u(-1, t) = 1$

![Comparison results between the reference solution and the predicted solutions by IFNN and PINN](image)

(a) reference solution    (b) predict solution by IFNN    (c) predict solution by PINN

(d) $t = 0.25$    (e) $t = 0.5$    (f) $t = 0.75$

Figure: Comparison results between the reference solution and the predicted solutions by IFNN and PINN
Numerical Experiments - 1D inviscid Burgers’ equation with shock wave (II)

IC: $\phi(x) = \begin{cases} 
2, & x \leq -1/2, \\
1, & -1/2 < x \leq 1/2, \\
0, & x > 1/2 
\end{cases}$, Neumann BC: $u'(-1, t) = u'(3/2, t) = 0$

Figure: Comparison results between the reference solution and the predicted solutions by IFNN and PINN
Numerical Experiments - 1D inviscid Burgers’ equation with rarefraction wave

IC: \( \phi(x) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases} \), Dirichlet BC: \( u(-1, t) = 0 \)

Figure: Comparison results between the reference solution and the predicted solutions by IFNN and PINN
Numerical Experiments - 1D inviscid Burgers' equation with sinusoidal initial data

IC: $\phi(x) = \sin(2\pi x)$, Periodic BC: $u(0,t) = u(1,t)$

(a) reference solution (b) predicted solution by IFNN (c) predicted solution by PINN

(d) $t = 0.25$ (e) $t = 0.5$ (f) $t = 0.75$

Figure: Comparison results between reference and predicted solutions by IFNN and PINN
Numerical Experiments - 2D inviscid Burgers’ equation with shock wave

Figure: Comparison results between the reference solution and the predicted solution by IFNN
By choosing $d = 1$ and $f(u) = u(1 - u)$ in (1) we get the Lighthill-Whitham-Richards (LWR) model

$$
\begin{aligned}
\begin{cases}
  u_t + [u(1 - u)]_x = 0, & x \in \mathbb{R}, t > 0, \\
  u(x, 0) = \phi(x), & x \in \mathbb{R},
\end{cases}
\end{aligned}
$$

(9)

where $u$ represents the density of cars on a road and $u \in [0, 1]$. When $u = 0$, there are no car on the road, and the road is completely full when $u = 1$. 
Numerical Experiments - LWR model for the traffic flow problem (traffic jams)

IC: \( \phi(x) = \begin{cases} 
1/2, & x < 0, \\
1, & x > 0,
\end{cases} \) Dirichlet BC: \( u(-1/2, t) = 1/2, \ t \in (0, 1] \)

Figure: Comparison results between reference solution and predicted solutions by IFNN and PINN

(a) Reference solution  (b) Predicted solution by IFNN  (c) Predicted solution by PINN

(d) \( t = 0.25 \)  (e) \( t = 0.5 \)  (f) \( t = 0.75 \)
Numerical Experiments - LWR model for the traffic flow problem (traffic light turning green)

IC: $\phi(x) = \begin{cases} 
1, & x < 0, \\
0, & x > 0,
\end{cases}$, Dirichlet BC: $u(-1, t) = 1$

Figure: Comparison results between reference solution and predicted solutions by IFNN and PINN
Conclusion remarks

Conclusion

- A fully-connected neural network with skip connection, “IFNN”, is proposed for solving the scalar conservation laws.
- The essential difference between our IFNN and PINN lies the choice of the loss function. IFNN takes the implicit form of the solution to formulate one of the essential terms for the loss function while PINN directly uses the residual of the original PDE.
- Extensive numerical experiments in 1D and 2D show that our IFNN is superior to PINN in capturing shock waves while their performance are comparable for the continuous solution cases.

Pros and Cons

- **Pros**
  
  the training of IFNN is much easier than that of PINN since it need not to use automatic differentiation for calculations of differential operators

- **Cons**
  
  IFNN requires the target PDEs to have the specific implicit form for their solutions, thus the scope of its feasibility may not be as wide as PINN
Thanks for your attention!