Abstract

- Discovering governing equations of a physical system, represented by partial differential equations (PDEs), from data is a central challenge.
- Current methods require either some prior knowledge (e.g., candidate PDE terms) to discover the PDE form, or a large dataset to learn a surrogate model of the PDE solution operator.
- We propose the first learning method that only needs one PDE solution, i.e., one-shot learning.
- We first decompose the entire computational domain into small domains, where we learn a local solution operator, and then find the coupled solution via a fixed-point iteration.

Problem setup: Learning solution operators of PDEs

Consider a physical system governed by a PDE defined on a spatio-temporal domain \( \Omega \subset \mathbb{R}^d \):
\[
\mathcal{F}(u(x); f(x)) = 0, \quad x = (x_1, x_2, \ldots, x_d) \in \Omega
\]
with suitable initial and boundary conditions. We define the solution operator as
\[
\mathcal{G} : f(x) \mapsto u(x).
\]

Dataset: \( T = \{(f_i, u_i)\}_{i=1}^{|T|} \), and \((f_i, u_i)\) is the \(i\)-th data point, where \(u_i = \mathcal{G}(f_i)\) is the PDE solution for \(f_i\).

Goal: Learn \(\mathcal{G}\) from \(T\), such that for a new \(f\), we can predict the corresponding solution \(u = \mathcal{G}(f)\).

Extreme difficult scenario: We have only one data point for training, i.e., one-shot learning with \(|T| = 1\), and we let \( T = \{(f_T, u_T)\} \).

- Assume we can select \(f_T\);
- We only predict \(f\) in a neighborhood of some \(f_0\), where we know the solution \(u_0 = \mathcal{G}(f_0)\).

Methods: One-shot learning based on locality

Idea: Consider that derivatives and PDEs are defined locally, i.e., the same PDE is satisfied in an arbitrary small domain inside \(\Omega\). We partition the entire domain \(\Omega\) into many small domains, i.e., a mesh of \(\Omega\).

Learning the local solution operator via a neural network.

Consider a mesh node at the location \(x^*\) (the red node). If we know the solution \(u\) at the boundary of \(\Omega\) \((\partial \Omega)\) and \(f\) within \(\Omega\), then \(u(x^*)\) is determined by the PDE. We use a neural network to represent this relationship
\[
\hat{\mathcal{G}} : \{u(x) : x \in \partial \Omega\} \cup \{f(x) : x \in \hat{\Omega}\} \mapsto u(x^*).
\]

Training dataset:
- “Large”: By traversing \(\Omega\) for all small local domains, we can generate many input-output pairs for training.
- “Diverse”: We choose \(f_T\) to be uniform random between -1 and 1 on each mesh node, i.e., \(f_T(x)\) is sampled from \(U(-1, 1)\).

Prediction via a fixed-point iteration.

For a new \(f = f_0 + \Delta f\), we use \(u_0\) as the initial guess of \(u\), and then in each iteration, we apply the trained network on the current solution as the input to get a new solution.

\[
\text{Initiate: } u(x) \leftarrow u_0(x) \text{ for all } x \in \Omega
\]
while \(u\) has not converged do
\[
\text{for } x \in \Omega \text{ do}
\]
\[
\hat{u}(x) \leftarrow \hat{\mathcal{G}}(\text{the inputs of } u \text{ and } f \text{ in } \hat{\Omega})
\]
\[
\text{update: } u(x) \leftarrow \hat{u}(x) \text{ for all } x \in \Omega
\]

Demonstration examples: Nonlinear diffusion-reaction equation

\[
\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} + ku^2 + f(x), \quad x \in [0, 1], \quad t \in [0, 1],
\]
with zero IC/BC. \(D = 0.01\) and \(k = 0.01\).

Training data for the diffusion-reaction equation:

<table>
<thead>
<tr>
<th>(i)</th>
<th>(j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i-1)</td>
<td>(j)</td>
</tr>
<tr>
<td>(i)</td>
<td>(j+1)</td>
</tr>
<tr>
<td>(i+1)</td>
<td>(j-1)</td>
</tr>
</tbody>
</table>

Prediction: We randomly sample \(\Delta f\) from a Gaussian random field (GRF): \(\Delta f \sim \mathcal{GP}(0, k(x_1, x_2))\), where the covariance kernel is \(k(x_1, x_2) = \sigma^2 \exp(-\|x_1 - x_2\|^2/2\ell^2)\).

References