开题报告

王一开

2022 年 1 月 5 日

1 选题背景和意义

在数据驱动的机器学习取得进展之前，许多工程和物理领域采用的都是物理模型驱动，这些物理模型大多以偏微分方程的形式刻画或描述，例如流体力学中的 Navier-Stokes 方程组、电磁场理论中的 Maxwell 方程组、量子力学中的 Schrödinger 方程组和相变问题中的 Allen-Cahn 方程等。通常情况下我们难以直接通过数学推导或者经验获得偏微分方程的精确描述，所以一般考虑求方程的近似解。数值方法是经典的偏微分方程解法，常用的数值方法主要有有限差分法、有限元法和有限体积方法，这些方法通过将给定的求解区域进行剖分，在剖分的网格上使用一系列基函数的组合进行逼近，最后得到方程的近似解。这些方法经过多年的发展，已经有了非常好的理论基础和实验结果，但是它们的求解严重依赖网格的剖分，网格划分的太粗求解精度会很低，而网格划分的太细又会带来巨大的计算代价和存储代价。除此之外，同一个数值方法求解不同的方程也有不同的过程，例如有限差分法求解不同的方程要考虑不同的网格划分格式，这个缺点也限制了数值方法的大规模应用。

随着数据和计算资源的爆炸式增长，机器学习与数据分析已经在多个领域例如计算机视觉，自然语言处理和生命科学产生了许多革命性的进展。然而很多时候在分析复杂的物理、生物或工程系统的过程中，数据采集的成本非常昂贵，我们不可避免地要面临在只有极其少量信息的情况下训练模型。在这种场景下，绝大多数先进的机器学习模型像深度神经网络、卷积神经网络等都会缺乏鲁棒性，而且模型的性能非常差。一部分研究人员考虑采用经典的机器学习算法去解决这些物理问题，经典的机器学习模型通常都是纯数据驱动的，通过建立一个从输入数据到输出数据的函数映射来训练一个有监督学习的机器学习模型，即依靠测量仪器收集到的历史数据学习一个具体的模型，模型性能的好坏与训练数据高度相关。然而这类纯数据驱动的算法却有很大的不足之处，第一纯数据驱动的算法缺乏解释性和适用性，针对不同的物理问题要考虑在模型中加入不同的物理限制，对于某个物理问题要设计针对该问题的特定模型去求解，而该模型无法应用于其他物理问题，这极大的限制了模型的适用范围；第二纯数据驱动的算法需要大量的训练数据使得模型收敛，而在许多物理和工程领域场景中，训练数据的获得代价非常昂贵，通常要面临在只有少量数据的情况下训练，此时训练出的模型往往泛化性能很差，无法准确预测物理参数；第三在物理和工程领域获得的数据常常隐含部分先验知识，如流体力学中的流场数据需要满足质量守恒和动量守恒定律，即满足 Navier-Stokes 方程组，而纯数据驱动的算法并为利用到这些知识，因此某种程度上并为充分利用训练数据。因此本文利用一种基于物理信息的神经网络 [4](Physics-informed Neural Network,PINN), 将数据驱动的机器学习和训练数据的先验知识结合在一起，能在少量训练数据的场景下，训练出自动满足物理约束的模型，在保证收敛的同时保持良好的泛化性能，能够准确预测需要的物理参数。

2 研究内容

在二十多年前就有科学家提出用神经网络求解常微分方程和偏微分方程 [2],[3]，但受限于当时的计算方法和计算资源，这一方法的效果并不是很好也并为得到足够的重视。近几年随着算力的增长，神经网络凸显出其对于非线性函数强大的逼近能力。布朗大学应用数学系的教授 Karniadakis 教授在神经网络的基础上，进一步将其于物理规律相结合提出了一套新的深度学习算法框架，叫做 “Physics Informed Neural Network（PINN）”，并首先将其用于求解偏微分方程的正问题与反问题，取得了很好的结果。PINN 提出后，大量基于这一框架的研究被展开，并逐渐产生了一门新的交叉领域———科学机器学习。

2.1 非线性偏微分方程

从数学函数逼近论的角度看，神经网络可以看作一个具有强大逼近能力的非线性函数，而偏微分方程的求解过程就是去寻找一个满足约束条件的非线性函数，因此可以考虑用神经网络去逼近偏微分方程的解。利用训练神经网络时广泛使用的自动微分 [1] 技术，可以将神经网络对它的模型参数和输入变量求微分从而将偏微分方程中的微分约束条件嵌入到神经网络的损失函数设计中，这样便获得了带有物理模型约束的神经网络，即基于物理信息的神经网络（PINN）。考虑一种一般形式的非线性偏微分方程

$$u_t + \mathcal{N}[u;\lambda] = 0 \; x \in \Omega, t \in [0,T] \tag{1}$$

$u(t,x)$ 表示该方程的解，$\mathcal{N}[u;\lambda]$ 是一个带有参数 λ 的非线性算子，x 是空间变量，t 是时间变量，Ω 是空间范围，T 是时间范围。传统的偏微分方程解法是给定 $u(t,x)$ 的初始状态、边界状态和参数 λ 的值，通过求解给定的方程来获得任意时刻和位置的 $u(t,x)$ 的函数值，如果无法求得方程的解析解，可以利用有限差分、有限元法等数值方法获得 $u(t,x)$ 在指定 (x,t) 处的数值解。

2.2 Burgers' 方程

以 Burgers' 方程为例，Burgers' 方程在许多领域都有应用，包括流体力学、空气动力学和交通流量等方面。带有 Dirichlet 边界条件的 Burgers' 方程有如下形式

$$\begin{aligned} &u_t + uu_x - (0.01)/\pi)u_{xx} = 0 \; x \in [-1,1], t \in [0,1] \\ &u(0,x) = -sin(\pi x), \\ &u(t,-1) = u(t,1) = 0. \end{aligned} \tag{2}$$

定义方程的左边为 $f(t,x)$:

$$f(t,x) := u_t + uu_x - (0.01)/\pi)u_{xx}$$

PINN 通过建立一个神经网络来逼近方程的解 $u(t,x)$, 为了更清晰的阐明该方法是如何实现的我们展示了部分 Python 代码，该方法使用当前最为流行的开源深度学习框架 Pytorch 实现。首先定义一个逼近 $u(t,x)$ 的神经网络:

```
def net_u(self, x, t):
        u = self.dnn(torch.cat([x, t], dim=1))
        return u
```

接着使用 Pytorch 的自动微分功能对该网络的输入变量 (x,t) 求微分从而将方程的微分约束形式使用神经网络表示出来。

```
    def net_f(self, x, t):
        """ The pytorch autograd version of calculating residual """
        u = self.net_u(x, t)

        u_t = torch.autograd.grad(
            u, t,
            grad_outputs=torch.ones_like(u),
            retain_graph=True,
            create_graph=True
        )[0]
        u_x = torch.autograd.grad(
            u, x,
            grad_outputs=torch.ones_like(u),
            retain_graph=True,
            create_graph=True
        )[0]
```

```
        u_xx = torch.autograd.grad(
            u_x, x,
            grad_outputs=torch.ones_like(u_x),
            retain_graph=True,
            create_graph=True
        )[0]

        f = u_t + u * u_x - self.nu * u_xx
        return f
```

PINN 的损失函数定义如下：

$$LOSS = LOSS_u + LOSS_f \tag{3}$$

其中 $LOSS_u$ 表示为

$$LOSS_u = \frac{1}{N_u}\sum_{i=1}^{N_u}|u(t_u^i, x_u^i) - u^i|^2 \tag{4}$$

$LOSS_f$ 表示为

$$LOSS_f = \frac{1}{N_f}\sum_{i=1}^{N_f}|f(t_f^i, x_f^i)|^2 \tag{5}$$

(4) 表示损失函数中数据驱动的部分，$\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ 表示通过初始状态和边界状态获取的训练数据，也包括通过实验采样、数值模拟等手段获取的训练数据。(5) 表示损失函数中物理模型约束部分，$\{t_f^i, x_f^i\}_{i=1}^{N_f}$ 表示对于 $f(x,t)$ 的训练配点，通过在时间和空间范围内采样 (x,t)，再利用自动微分技术便可高效计算出 $LOSS_f$。我们期望训练出的神经网络不但与训练数据的误差要尽可能小，而且要尽可能满足方程的约束条件，即 $LOSS_u$ 和 $LOSS_f$ 要尽可能接近 0。

2.3 实验结果

我们分别用传统方法和神经网络做了实验来进行对比。

2.3.1 有限元方法

传统方法使用有限元法来及求解偏微分方程的数值解，我们使用 Fenics 来实现，Fenics 是一个开源的求解偏微分方程的计算平台，可以快速有效的将物理模型转化为有限元代码。在空间上我们划分了 255 个区域，时间上划分了 100 个区域。图 1 展示了使用有限元求解的结果。

3 神经网络

4 研究方法

5 研究计划

参考文献

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018.

[2] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[3] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

图 1: 使用 Fenics 求解

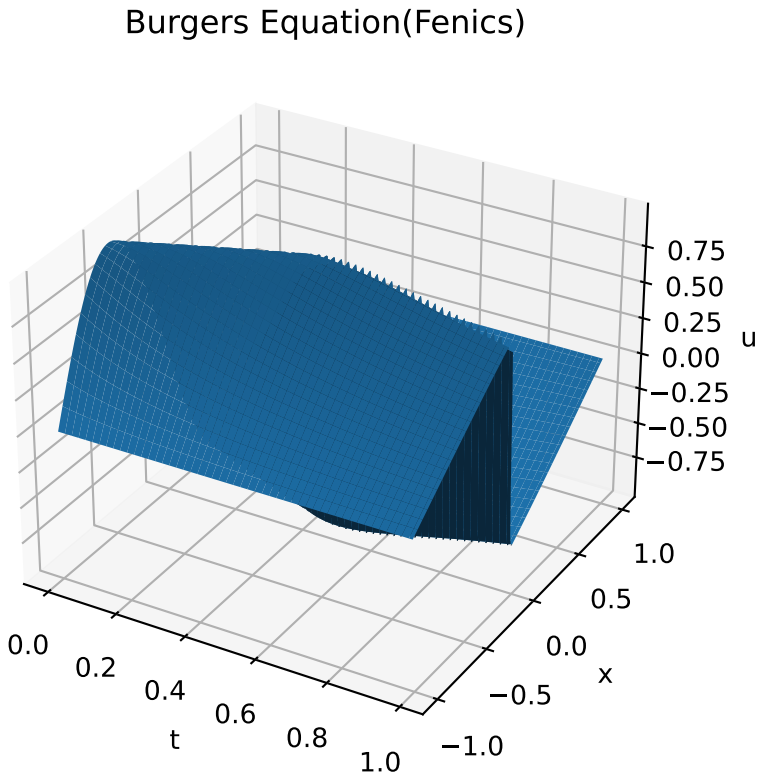


图 2: 训练数据分布

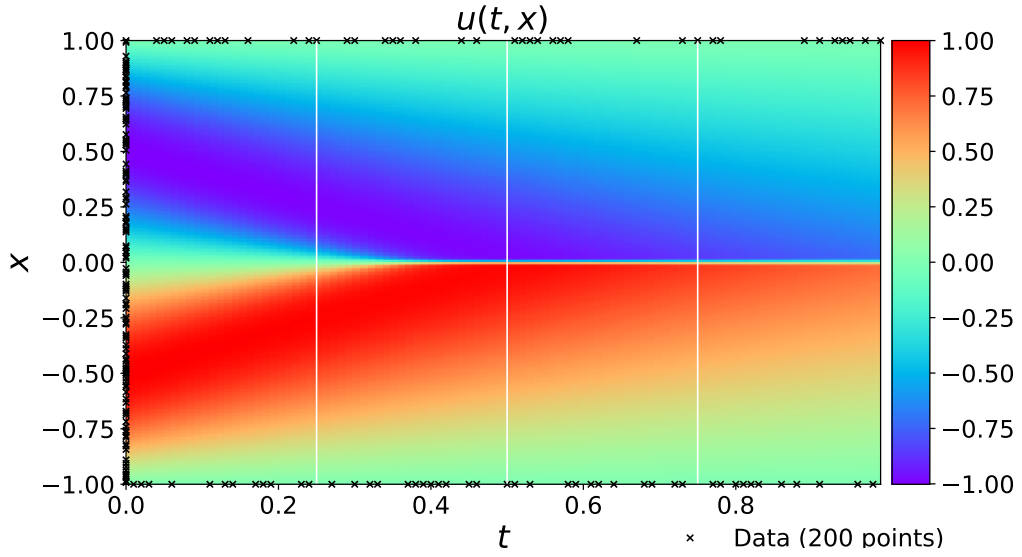
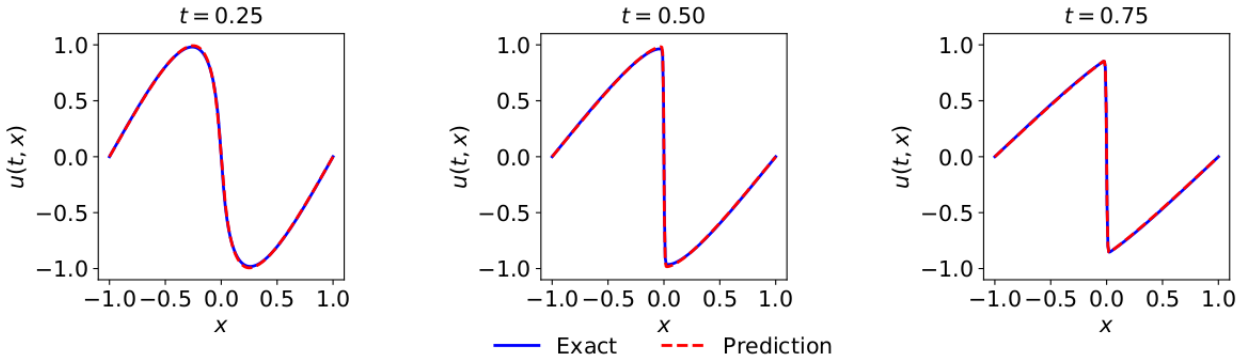


图 3: 结果对比



[4] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.