arXiv:2303.02890v1 [cs.LG] 6 Mar 2023

物理信息神经网络分析

A dissertation submitted to the University of Manchester
for the degree of Master of Science
in the Faculty of Science and Engineering

2021

Edward Small数学系

目录

List 图 i v

表目录 vii

算法目录 viii

摘要 ix

声明 x

知识产权声明 xi

符号说明 xii

缩略语 xiii

致谢 xiv

1 引言 1

1.1 什么是人工神经网络？.2 1.1.1 网络.2 1.2 什么是偏微分方程？.4

2 Tr训练网络以近似函数 6

2.1 优化算法.7 2.1.1 随机梯度下降.8 2.1.2 ADAM 优化.8 2.1.3 L-BFGS 优化.9 2.1.4 非精确线搜索.10 2.1.5 验证.11

3 偏微分方程神经网络 12

3.1 物理信息神经网络的损失函数.1 3

3.2 神经网络的微分.14 3.2.1 微分技术的问题.15 3.2.2 自动微分.18 3.3 通用逼近器.22

4 SimplePINNExample 25

4.1 通用框架.25 4.2 一维波动方程.26 4.2.1 解析解.26 4.2.2 神经网络.27 4.3 误差分析.27 4.3.1 误差测量.28 4.3.2 架构设计.29 4.3.3 解收敛.31 4.4 改进学习过程.33 4.4.1 误差传播.33 4.4.2 误差模式.36

5 当前应用与研究领域 42

5.1 地球物理层析成像.42 5.1.1 波场重建反演.42 5.1.2 使用物理信息神经网络求解波动方程 45 5.1.3 反问题.49

6 结论 55

参考文献 57

A

64A.1 证明与示例.64 A.1.1 表达式膨胀.64 A.1.2 一维波动方程的解.65 A.2 误差与绘图代码.66

图目录

1.1 接收 n 个输入的单个感知器的可视化表示 . . . . . 2 1.2 一个具有 $x \in \mathbb{R}^4$ 个输入、2 个隐藏层和一个输出层的人工神经网络 $\hat{y} \in \mathbb{R}^3$ 32.1 动量如何影响下降方向的可视化表示 . . . . 9 3.1 对热方程进行有限差分近似，完成 3.7 中的示例。从左到右： $t = 0$、 $t = 10$、 $t = 20$.16 3.2 使用人工神经网络近似估计一个简单常微分方程的解。近似解极其精确，一阶导数也是如此。尽管如此，使用有限差分法估计二阶导数时，可以看出与实际二阶导数相比，我们会损失多少光滑性.18 3.3 方程 3.17 的计算图表示，展示如何将 2 个变量共同处理并进行运算，以生成任意输出 . . . 20 3.4 方程 3.17 的计算图表示，这次存储该数值.21 3.5 具有狄利克雷边界条件且无阻尼的简单振动膜的级数解，如 3.19 中所述.23 4.1 4.1 中所述波的行为。由于不存在阻尼，该波将始终如此振荡，且周期为 4 . . . 26 4.2 通过物理信息神经网络近似得到的波的行为。对于上述示例，网络结构为 $[64, 32, 16, 8],$，因此具有按层递减的 4 个隐藏层.28 4.3 波的行为的两种可能的简单划分。划分 1 将数据分成导数性质发生变化的区段，划分 2 则根据数值接近 -1、0 还是 1，将数据子集分类。划分 1 意味着使用 8-4-2 网络，而划分 2 意味着使用 16-8-4-2 网络.30

4.4 真实解与 8-4-2 网络（所测试的最简单网络）之间的误差.32 4.5 真实解与 64-32-16-8 网络之间的误差 . . . . . 33 4.6 随着最小化的进行， L_2 误差的收敛情况 . . . . . 34 4.7 64-32-16-8 网络的 $\Lambda(x,t)$ 收敛行为。每幅图均显示相对于真实解的绝对误差.35 4.8 对于取自 $t=2$ 而非直接取自初始条件的数据， $\Lambda(x,t)$ 的收敛行为.36 4.9 二维区域数据集的一种可能的数据分组方式。 $X^{(i)}$ 的每次迭代都会扩展每组数据周围的可访问域。该算法会先对各个分区内的行为进行建模，再扩展到区域其他部分的行为。 . . . . . 37 4.10 来自不同定义域位置的数据集对应的近似解和误差模式。该模式与数据分布无关.38 4.11 $t=0$ 处所有数据对应的解析解导数与物理信息神经网络解导数之间的误差.39 4.12 伯格斯方程的真实解和物理信息神经网络近似解 40 4.13 上：解析解与物理信息神经网络近似解之间的误差。中：空间导数。 $x\approx 0$ 处的导数远大于其他所有导数。下：时间导数。其数值显著小于空间导数的极值.41 5.1 使用地震波成像的基本示意图。波（红色）从震源 q_s穿过介质 m （大多数情况下为地球）传播，传感器 d_r 在地表测量波响应。响应会随介质结构而变化.43 5.2 求解波动方程的网络架构。该网络由 10 层组成，共有 1024 个节点。最终输出层采用线性激活，而非非线性激活函数.46 5.3 描述介质速度的不同数据集。白点表示产生声波的震源.47 5.4 使用三种方法（有限差分、人工神经网络、物理信息神经网络）得到的分层数据集波动行为估计。人工神经网络未能捕捉较大 t 处的许多反射波行为，但能对主导波动行为进行建模。 48

5.5 有限差分模拟与两个网络各自的差异。物理信息神经网络与有限差分模拟之间的误差几乎可以忽略。48 5.6 左：真实 Marmousi 速度模型。右：物理信息神经网络波动方程反演算法的初始速度模型.52 5.7 左上：使用 3 赫兹迭代 1 次后的物理信息神经网络速度模型。右上：使用 3 赫兹迭代 4 次后的物理信息神经网络速度模型。左下：使用 4 赫兹迭代 4 次后的物理信息神经网络速度模型。右下：使用 5 赫兹迭代 3 次后的物理信息神经网络速度模型.53 5.8 左：使用全波形反演从分层初始速度模型估计得到的速度模型。右：使用全波形反演从物理信息神经网络解初始速度模型估计得到的速度模型.53

表目录

3.1 此处列出了图 3.4 中计算图的中间变量的数值。利用链式法则，一次前向传播即可计算给定点处函数的数值及其梯度。计算机并不会真正“知道”切向量的表达式——这些数值是使用对偶数时的副产品。随后利用这些数值和链式法则计算导数. 22

4.1 不同神经网络结构在各种误差范围内的误差表。大多数架构即使可训练参数数量很少，也能对解给出合理的估计值。然而，在英特尔四核 i7 上训练一个含有 4417 个参数的系统耗时超过一小时（30000 次迭代），而含有 73 个参数的系统则不到一分钟。这就是为什么找到足够小的网络至关重要。31

算法目录

1 ADAM优化.9 2 BFGS算法.10 3 基于物理信息神经网络的波动方程反演.51

摘要

尽管描述我们所处世界动态的偏微分方程数百年来一直受到深入研究，但要在复杂的高维条件和区域中求解这些方程，仍然是极具挑战性的数学与计算难题。解析方法的应用可能十分繁琐，而数值方法则可能带来误差和不精确性。此外，有时我们缺乏足够的信息或知识，无法恰当地构建问题，因而难以应用这类方法。

本文提出了一种近似求解物理系统的新方法——物理信息神经网络。文中介绍人工神经网络的概念，定义目标函数，并探讨优化策略。随后，将偏微分方程作为约束条件纳入优化问题的损失函数，使网络能够利用其所建模的物理系统的动态信息。

本文展示了一些直观的示例，并探讨了更复杂的应用，以展现物理信息神经网络的强大能力，例如在地震成像中的应用。本文分析了解误差，并提出了改善收敛和／或提高解的精度的建议。结论部分也谈到了相关问题和局限性，并探讨了物理信息神经网络最适用的领域，以及未来的发展方向。

声明

本学位论文所涉及的工作，均未曾用于支持申请本校或任何其他大学或其他教育机构的其他学位或资格。本文件在提交后重新排版，但内容保持不变。

知识产权声明

i. 本学位论文（包括其任何附录和/或附表）的作者拥有其中某些版权或相关权利（“版权”），并已授予曼彻斯特大学使用此类版权的某些权利，包括用于行政管理目的。

ii. 本学位论文的全部或摘录副本，无论是纸质版还是电子版，均仅可依照经修订的1988年《版权、外观设计和专利法》及其项下颁布的法规制作；或在适当情况下，依照曼彻斯特大学已签订的许可协议制作。任何此类副本均必须包含本页。

iii. 某些版权、专利、外观设计、商标及其他知识产权（“知识产权”），以及学位论文中受版权保护作品的任何复制品（例如图表和表格，以下简称“复制品”），可能并不归作者所有，而是归第三方所有。本学位论文中描述的此类知识产权和复制品，未经相关知识产权和/或复制品所有者事先书面许可，不得提供他人使用。

iv. 关于本学位论文的披露、出版和商业化，以及其中所述版权、任何知识产权和／或复制品可在何种条件下进行的更多信息，请参阅《大学知识产权政策》（见http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487）、存放于大学图书馆的任何相关学位论文限制声明、大学图书馆条例（见http://www.manchester.ac.uk/library/ab- outus/regulations），以及《大学学位论文呈现指南》。

符号说明

α, β, γ, ... 用于表示标量的小写希腊字母 $\mathbb{R} a, b, x, y, z$ 用于表示向量 $\mathbb{R}^n x_i$ 向量中的一个一般元素 $x x_k$ 向量的状态 x 在算法的第 k 次迭代中 $\hat{}y$ 真实输出的估计值 yA, B, C, W, M 用于表示矩阵 $\mathbb{R}^{n \times m}$ 或集合 $A_{i,j}$ 矩阵中的一个子矩阵 Af, g, ϕ, σ 将一个集合映射到另一个集合的函数 f_t, ϕ_{yy}, u_x 对下标所指变量求导后的函数 f_n, ϕ_n 级数或集合中的第 n 个函数 F, G 小写函数的人工神经网络近似 f, gL 网络中的层数 l 网络中的特定一层 N 集合或级数中的元素个数 $n^{[l]}$ 特定层中的特定神经元

$\mathcal{L}$ 损失函数 $||x||_p$ 向量范数 $\left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$

$\mathcal{J}$ 雅可比矩阵 $\mathcal{H}$ 海森矩阵 $\mathcal{N}$ 微分算子

Λ 物理信息神经网络 θ 用于参数化人工神经网络的数值，例如权重和偏置

缩略语

AD 自动微分 ADAM 自适应矩估计 ANN 人工神经网络 AutoGrad 自适应随机梯度下降 BC 边界条件 CPU 中央处理器 FD 有限差分 FEA 有限元分析 FWI 全波形反演 GPU 图形处理器 IBM 国际商业机器公司 IC 初始条件 KdV Korteweg–De Vries ML 机器学习 MRI 磁共振成像 MSE 均方误差 NLP 自然语言处理 L-BFGS 有限内存Broyden–Fletcher–Goldfarb–Shanno算法 LU LU分解 ODE 常微分方程 PDE 偏微分方程 PINN 物理信息神经网络 RMSProp 均方根传播 SAS 统计分析软件 SGD 随机梯度下降 s.t. 使得 TV 总变分 WRI 波场重建反演 w.r.t 关于

致谢

谨向我的导师 Oliver Dorn 博士致以诚挚的谢意，感谢他在撰写本论文期间给予我丰富的经验与智慧。在我于曼彻斯特大学求学的大部分时间里，他都是激励我的重要源泉；他的指导对我开展的研究至关重要。我对他所讲授的数值优化与反问题课程十分喜爱，这也正是最初促使我研究这一课题的原因。

当然，我也要感谢那些一直陪伴在我身边的人。无论是母亲 Annette，在我沉思时偶尔为我送来一杯咖啡；父亲 Graham，总是愿意花时间倾听我；还是朋友 Matt，时常提醒我休息一下也未尝不可，我都将永远心怀感激。我尤其感激他们在我兴致勃勃地讲述研究内容时，尽管并不真正明白我在说什么，仍会点头附和。

最后，我还要感谢一位特别重要的人。虽然她无法陪在我身边，却始终给予我无尽的支持——我的伴侣 Jess。过去两年来，澳大利亚从未显得如此遥远，但希望我们很快就能相见。

第1章

引言

机器学习是数学中一个相对年轻且发展迅速的分支。虽然直到 1952 年，IBM 计算机科学家 Arthur Samuel 才首次提及机器学习（当时他正在开发复杂算法，让计算机能够高效地玩跳棋 [1]），但机器学习的基础散见于近代历史之中，其开创离不开许多知名数学家的贡献，例如拉普拉斯、马尔可夫 [7, 10], 贝叶斯 [6], 和图灵 [9], 等。它提出了一个优雅的问题：“如果我的计算机能够学习，会怎样？”也就是说，如何创造一种算法，使其解决问题的方式具有动态性——能够从错误中学习，并通过反复迭代不断调整和进化，从而变得更加准确?

现代数学家数十年来不断改进这些学习过程。随着大数据[1], 时代的到来，这些算法得到了令人难以置信的应用。从人脸识别软件 [11], 到自动驾驶汽车 [12] ，再到定向广告 [13], ，机器学习算法已经处理了海量数据，从而为人类从未能解决的问题提供了明确的解答。

机器学习的一个子分支——人工神经网络（ANNs）——近期的一项进展，是将系统已知的物理学作为优化学习过程的约束条件，最终求得偏微分方程（PDEs）的近似解。尽管这一过程较为复杂，但它不仅能够带来更准确的结果，还能以更高的计算速度实现这些结果。这类特殊的网络约束被称为物理信息神经网络（PINNs）。

[1]全球最大的数据管理和分析公司之一 SAS 将大数据定义为规模极大、速度极快（每个时间步长都会产生大量新数据）或复杂度极高（可变性很高且真实性高）的数据集，以至于无法通过传统统计方法分析这些数据。这类数据在处理和存储方面带来了独特的挑战。

1.1 什么是人工神经网络?

PINNs 是 ANNs 的一种，因此在了解 PINNs 之前，掌握 ANNs 的基本知识非常重要——包括它们如何处理数据、生成结果，以及如何学习。

1.1.1 网络

顾名思义，人工神经网络的设计深受神经元基本运算及其原理的启发。神经元是一种神经细胞，也是神经系统的基本构成单元。单个人工神经元称为感知器[2]。感知器接收一组输入 $x \in \mathbb{R}^n$ 并产生一个输出 $y \in [-1, 1]$，其中 y 用于衡量感知器的“活跃程度”。感知器通过对输入执行以下 3 个基本步骤来实现这一点：

1. 将每个输入 $x_i \in x$ 与其对应的权重 $w_i \in w$相乘，然后求和。每个权重 w_i 可以理解为感知器对各个输入 x_i的敏感程度。例如，如果 $w_i = 0$ ，那么输入 x_i 不会影响感知器的激活。

2. 将偏置 β 加到这个总和上。这个偏置可以理解为：当 $x_i = 0$ 为 $i = 1$、.....、n时，神经元的活跃程度。

3。将该数值输入激活函数 σ： $\mathbb{R} \to [-1, 1]$，例如 tan h.

因此，感知器 a 的激活可以表示为

$$a = \sigma\left(\sum_{i=1}^{n}(x_i w_i) + \beta\right) \tag{1.1}$$

其视觉呈现如图1.1所示。

因此，直观地说，人工神经网络 F （近似于 f）就是由这些感知器构成的集合，它们彼此连接 [14], ，前面神经元的激活会影响

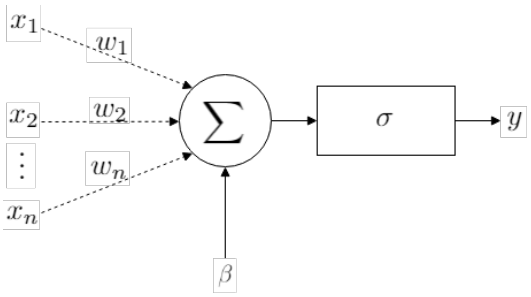


图1.1：单个感知器接收 n 输入的视觉表示。

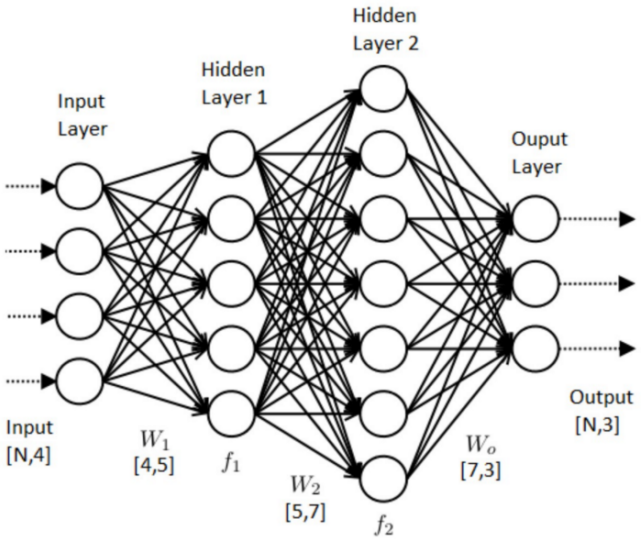


图 1.2：一个包含 $x \in \mathbb{R}^4$ 个输入、2 个隐藏层和一个输出层的人工神经网络 $\hat{y} \in \mathbb{R}^3$

其他神经元的激活，使得 $F: \ x \to \hat{y}$。通常，人工神经网络会组织成$l = 1$，2，..... L 层，每层包含 $n^{[l]}$ 个神经元。输入层 $n^{[1]}$ 与输出层 $n^{[L]}$ 之间的任何层通常称为隐藏层，因为用户无法访问其状态 [15]。用户只能访问输入层和输出层。

因此，对于 $l = 2, ..., L-1$，激活值 $a^{[l]}$ 为

$$\begin{bmatrix} a_1^{[l]} \\ a_2^{[l]} \\ \vdots \\ a_{n^{[l]}}^{[l]} \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n^{[l-1]}} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n^{[l-1]}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n^{[l]},1} & w_{n^{[l]},2} & \dots & w_{n^{[l]},n^{[l-1]}} \end{bmatrix} \begin{bmatrix} a_1^{[l-1]} \\ a_2^{[l-1]} \\ \vdots \\ a_{n^{[l-1]}}^{[l-1]} \end{bmatrix} + \begin{bmatrix} b_1^{[l]} \\ b_2^{[l]} \\ \vdots \\ b_{n^{[l]}}^{[l]} \end{bmatrix} \right) \tag{1.2}$$

可以简化为

$$a^{[l]} = \sigma(W^{[l]} a^{[l-1]} + b^{[l]}) \tag{1.3}$$

其中 $a^{[l]}, b^{[l]} \in \mathbb{R}^{n^{[l]}}$，且 $W^{[l]} \in \mathbb{R}^{n^{[l]} \times n^{[l-1]}}$。这就是人工神经网络如何在系统中传播信息以找到 f 的映射的关键：$x \to y$。

神经元可以通过多种不同的方式连接，从而构成特殊类型的网络，例如卷积网络[2] 或循环网络[3]。图 1.2 展示了一个全连接人工神经网络的示例，因为第 $l-1$ 层中的每个节点都与第l层中的每个节点相连。为简单起见，除非另有说明，我们主要讨论全连接的前馈人工神经网络。

[2]卷积网络具有局部性或分组特性，即输入数据中并非每一层的每个神经元都与下一层的每个神经元相连接。[3]循环网络会保留一些先前输入和输出的内存，因此先前的条目可以影响未来的结果。

人工神经网络的一个重要结论是，具有单个隐藏层的网络 F 可以在一个区域 I_n （该区域为 n 维）内以任意精度 $\epsilon > 0$ 逼近任意连续函数 f ，使得

$$|F(x) - f(x)| < \epsilon \qquad \forall \quad x \in I_n \tag{1.4}$$

前提是对 W 和 b （人工神经网络 F的权重和偏置）的数值不设限制。因此，精度取决于神经元的数量。这被称为通用逼近定理[8]。

1.2 什么是偏微分方程?

偏微分方程（PDE）是一种通用方程，用于描述多变量函数 [16]中变量变化率之间的关系。一个著名的例子是一维热方程，它描述了具有一个空间维度的杆上的热通量 x随时间 t的变化，即

$$u_t = ku_{xx} \tag{1.5}$$

其中， $u_t = \frac{\partial u}{\partial t}$ 表示温度随时间的变化， $u_{xx} = \frac{\partial^2 u}{\partial x^2}$ 表示温度在空间中的变化，而 k 是一个对流项，用于描述能量在杆中流入或流出的难易程度 [17]。该方程将特定空间点和时间点处温度随时间的变化，与特定空间点和时间点处温度在空间中的变化联系起来。

偏微分方程通常用于描述各种物理系统，例如声学、扩散、电磁学、流体动力学，甚至量子力学。 [18], 因此，求解这些方程（给定某些初始条件（初始条件）[4] 和边界条件（边界条件）[5]）是一项极其重要的数学工作。遗憾的是，由于多种原因，要为真实物理现象求解偏微分方程可能极具挑战性。

解主要分为两种：解析解和数值解。解析解可以理解为闭式解，既满足偏微分方程，也满足问题的初始条件和边界条件；有时这样的解是唯一的（取决于问题的定义方式）。解析求解偏微分方程可能会耗费大量时间，需要先对偏微分方程进行分类，再运用周期延拓、傅里叶级数 [19], 和散射 [20]等复杂方法。即便如此，也无法保证所得的解易于使用，而且有时通解会以复积分或

[4]如果我们知道初始条件，就能知道函数在时间 $t = 0$[5]所有变量上的总体形态。如果我们知道边界条件，就能知道在感兴趣区域的极端部分，函数在所有时间 t 的行为。

无穷级数的形式给出

$$\phi(x,t) = \sum_{n=1}^{\infty} \alpha_n \phi_n(x,t) \tag{1.6}$$

其中每个 ϕ_n 都是偏微分方程的一个解。因此，即使 $\alpha_n \to 0$ 以 $n \to \infty$ 的速度迅速变化，我们也必须接受，我们只能以任意精度估计真实解。

在现实世界建模中，求解析解也会因其他原因而极具挑战性。这些挑战包括：不知道问题的确切初始条件和边界条件，数据集过于稀疏，或者区域过于复杂，无法以可用的方式准确定义。当人们考虑到某些广泛使用的偏微分方程（例如描述流体运动的著名纳维–斯托克斯（NS）方程）实际上尚未得到充分理解这一事实时，问题就会严重得多。事实上，我们甚至不知道三维流体流动是否总是存在光滑解，而湍流和奇异性仍然是一个巨大的难题 [21]。由于纳维–斯托克斯方程支配着大气动力学等物理现象的演变，进而影响天气预测，因此不难理解为什么研究这些方程至关重要。

数值方法可以对此提供帮助，例如有限差分（FD）和有限元分析（FEA）。有限差分通过估计导数的数值，来估计函数在空间和时间中下一个点处的数值；而有限元分析则使用变分方法，将边界条件的近似解分布到各个更容易求解的元素上，然后通过插值将这些解拼接起来。

这些技术本身也可能存在一些问题。例如，有限差分方法中的误差会传播到下一次迭代中，因此，将其用于较大区域和较长时间范围时会有一定挑战 [22]。有限元分析在跨越元素时可能存在连续性问题，这种现象称为通量跳跃，并会导致能量损失 [23]。在复杂区域中，使用任一方法获得准确的解通常都需要很强的计算能力；如果我们想知道空间和时间中某个特定点处的函数数值，通常还需要知道该点周围其他空间和时间点处的数值——而解析解则不需要这样做。

第2章

训练网络以逼近函数

人工神经网络的目标可能是复现某个未知函数的行为 f :$\mathbb{R}^n \to \mathbb{R}^m$。在机器学习中，主要有三种学习过程

1. 监督学习。
2. 无监督学习[1]。
3. 强化学习[2]。

当有可用的带标签数据集时，可以使用监督学习 [25]。也就是说，虽然真实函数 f 可能未知，但我们确实可以获得一组输入$X \in \mathbb{R}^{n\times p}$ 以及与之匹配的输出 $Y \in \mathbb{R}^{m\times p}$ ；其中，如果 $x \in X$ 是 X 中的一列，而$y \in Y$是 Y中对应的一列，那么

$$f(x) = y \tag{2.1}$$

如果用于近似 f 的人工神经网络称为 F，那么我们可以说，对于数据集 $x \in X$ ，ˆ F 的估计输出称为ˆ$y \in Y$，使得

$$F(x) = \hat{y} \tag{2.2}$$

因此，人工神经网络的目标是在所有输入和输出上最小化损失函数 [56] $\mathcal{L}$

$$\min_{\theta} \mathcal{L}(\theta) = \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\text{loss}}(y, F(x, \theta)) \tag{2.3}$$

[1]当已知函数的输入，但不知道对应的输出时，会使用无监督学习。因此，其目标通常是找出输入数据中的类别、分组或结构。这种学习类型常用于降维[2]当网络通过惩罚/奖励机制进行学习时，会使用强化学习。给定一个目标，网络应进行自我优化，以用最小的惩罚获得最大的奖励

其中， θ 是决定 F行为的可训练参数，例如权重和偏置。衡量损失的方法有很多种（合页损失、对比损失、熵等），但最容易理解的是最小化输出的欧几里得范数平方 [26], ，因此

$$\begin{aligned}\mathcal{L}_{\text{loss}}(y, F(x,\theta)) &= \sqrt{(y_1-\hat{y}_1)^2+(y_2-\hat{y}_2)^2+...+(y_n-\hat{y}_n)^2}^2 \\ &= ||y-\hat{y}||_2^2\end{aligned} \tag{2.4}$$

这意味着，当不存在任何能够改变参数 θ 并减小相同输入下函数输出与人工神经网络输出之差的大小的变更时，便找到了最小值。

显然，寻找一组使函数达到最小值的参数 θ 颇为复杂。如果一个人工神经网络有 L 层，每层中最大的层包含 m 个节点，并连接到 n 个节点，且具有 $n \geq m$，那么每次通过网络的传播大约为 $\mathcal{O}(L(nm+n))$。我们知道 $L \leq n$，因此，如果 $m = n$，则一次前向传播的规模为 $\mathcal{O}(n^2)$，因为最大的开销是矩阵-向量乘法（每条连接都要执行一次）。例如，对于图 1.2 所描述的简单人工神经网络，从隐藏层 1 到隐藏层 2 本身就需要进行 42 次计算，因此很容易看出，随着人工神经网络规模的增大，这一开销会急剧增加。

直观地说，如果出现 $L = n$ 的情形，那么每一层中只能有一个神经元，因此我们有 n 次标量乘法和加法，最终仍然达到 $\mathcal{O}(n^2)$。

训练网络是大部分开销产生的地方。在最坏情况下，我们可能需要使用 n 个数据点进行 n 次迭代，以调整 n^2+n 个权重和偏置，时间复杂度为 $\mathcal{O}(n^4)$。因此，在设计和训练人工神经网络时采用智能策略非常重要。

2.1 优化算法

如今，有许多方法可用于最小化公式2.3。对于线性问题，共轭梯度法可以在 n 步内找到最小值，其中 n是输入数据 [27]的维数。不同方法所需的函数信息各不相同，例如雅可比矩阵[3] 和/或海森矩阵[4], ；不过，根据对函数的了解，可以采用不同策略以数值方式求得最小值。有些方法甚至会使用海森矩阵估计和/或矩阵逆，从而加快计算速度 [28]。

[3]如果 $x \in \mathbb{R}^n$ 是输入，且 $f: x \to y \in \mathbb{R}^m$，那么 f 的雅可比矩阵 $\mathcal{J} \in \mathbb{R}^{m \times n}$ 是一个矩阵，其中包含 f_i x_j $\mathcal{J}_{ij} = \frac{\partial f_i}{\partial x_j}$ 的每个输出相对于每个输入的偏导数；因此，如果 $x \in \mathbb{R}^n$ 是输入，且 $f: x \to y \in \mathbb{R}$，那么 f 的海森矩阵 $\mathcal{H} \in \mathbb{R}^{n \times n}$ 是一个矩阵，其中包含 f 关于每对输入的二阶偏导数和混合偏导数 $\{x_i, x_j\}_{i,\ j=1}^{n}$，因此 $\mathcal{H}_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$

[4]

非线性问题（例如最小化人工神经网络的损失函数）要复杂一些，但也存在适用于这类问题的数值方法（如高斯–牛顿法） [29]。其中一些方法包括随机梯度下降 (SGD) 、有限内存 *Broyden–Fletcher–Goldfarb–Shanno* (L-BFGS) 算法（这是一种二阶拟牛顿法，它利用二阶导数（海森矩阵）逆的估计值来选择合理的搜索方向），以及 ADAM 优化器（它结合了两种最小化方法以加快学习）。

2.1.1 随机梯度下降

普通梯度下降通过初始化一组随机权重和偏置θ_0 ，并在每次迭代中计算损失函数相对于所有参数在所有数据点上的梯度来进行。然后，我们沿此方向更新 θ ，移动的幅度可以是预先选定的数值，称为训练参数 γ；也可以是需要计算得出的更具体数值，称为步长 α [30]。

$$\theta_{k+1} = \theta_k + \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \mathcal{L}(x_i; \theta_k) \tag{2.5}$$

然而，如果数据集 n 非常大，那么每一步的计算开销都会很高。为了降低这一开销，我们可以将数据划分为若干子集，并改为在每个较小子集上随机进行最小化，这种方法称为随机梯度下降（SGD） [31]。这会显著提高迭代速度。SGD主要有两种类型

1. 全量随机梯度下降，即每次迭代都针对单个数据点 $x_i \in x$最小化目标函数

$$\theta_{k+1} = \theta_k + \gamma \nabla_\theta \mathcal{L}(x_i; \theta_k) \tag{2.6}$$

2. 小批量处理，即每次迭代针对 $n_s \ll n$ 个数据点进行最小化，因此

$$\theta_{k+1} = \theta_k + \gamma \frac{1}{n_s} \sum_{i=1}^{n_s} \nabla_\theta \mathcal{L}(x_i; \theta_k) \tag{2.7}$$

小批量处理可以随机进行，使得每次迭代都会随机生成 x 的一个子集；也可以在训练前将数据划分为不同的子集。

2.1.2 ADAM 优化

自适应矩估计（ADAM）算法将自适应梯度下降（AdaGrad）与均方根传播 （RMSProp） [32]相结合。Adam 背后的主要驱动力在于，下降方向会考虑当前动量 [33], ，而步长经过谨慎生成，以创建一个信赖域；在信赖域之外，我们无法确定目标函数的形状。ADAM（算法

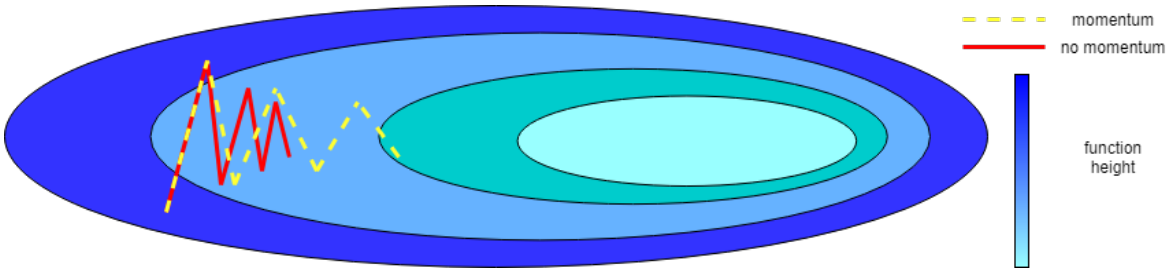


图2.1：动量如何影响下降方向的示意图

算法 1 ADAM 优化

要求： α ▷ 步长
要求： ϵ ▷ 收敛阈值
要求： $\beta_1,\ \beta_2 \in [0,1)$ ▷ 动量估计的指数衰减率
要求： $\mathcal{L}$ ▷ 目标函数
要求： θ_0 ▷ 初始参数
$m_0 \leftarrow 0$ ▷ 1一阶零向量
$v_0 \leftarrow 0$ ▷ 2二阶零向量
$i \leftarrow 0$ ▷ 设置迭代计数器
$\delta \leftarrow 10^{-8}$ ▷ 防止除以零
当 $\mathcal{L}(\theta_i) > \epsilon$ **时**
$i \leftarrow i+1$ ▷更新迭代计数器
$\mathcal{J}_i \leftarrow \nabla_{\theta_{i-1}}(\theta_{i-1})$ ▷ 计算梯度
$m_i \leftarrow \beta_1 m_{i-1} + (1-\beta_2)\mathcal{J}_i$ ▷ 更新有偏一阶估计
$v_i \leftarrow \beta_2 v_{i-1} + (1-\beta_1)\mathcal{J}_i \odot \mathcal{J}_i$ ▷ 更新有偏二阶矩估计
$\hat{m}_i \leftarrow \frac{m_i}{1-\beta_1^i}$ ▷ 偏差校正一阶估计
$\hat{v}_i \leftarrow \frac{v_i}{1-\beta_2^i}$ ▷ 偏差校正的二阶估计
$\theta_i \leftarrow \theta_{i-1} - \alpha \frac{\hat{m}_i}{\sqrt{\hat{v}_i}+\delta}$ ▷ 更新参数
结束循环
返回 θ_i

1）通过使用梯度和梯度平方的移动平均值来计算动量，其中超参数 β_1 和 β_2 控制指数衰减。

超参数的常用数值为 $\beta_1 = 0.9$ 和 $\beta_2 = 0.999$，其中 $a \odot b$表示向量 a 和 b之间的逐元素乘法。步长受限于当前点附近的一个区域；在该区域之外，我们缺乏足够的信息，无法准确计算梯度。

2.1.3 L-BFGS 优化

对于包含大量参数的大规模问题，务必要牢记，我们的内存并非无限。参数数量越多，雅可比矩阵 $\mathcal{J}$ 和海森矩阵 $\mathcal{H}$ 就越大。L-BFGS 在估算第 k 步 $\mathcal{H}_k$ [34]的海森矩阵时，只考虑过去的 m 次更新，从而限制内存使用量。

算法 2BFGS算法

要求： θ_0 ▷ 最优参数的初始猜测
要求： ϵ ▷ 最小化的容差
要求： $\mathcal{H}_0$ ▷ 海森矩阵的初始估计（通常为 I）
要求： $\mathcal{L}$ ▷ 要最小化的目标函数
$k \leftarrow 0$
当 $\mathcal{L}$（θ_k）$> \epsilon$ **时**
$p_k \leftarrow -\mathcal{H}_k^{-1}\nabla\mathcal{L}(\theta_k)$ ▷ 求搜索方向
$\alpha_k \leftarrow$ 最小化$_\alpha\ \mathcal{L}(\theta_k + \alpha p_k)$ ▷ 执行线搜索（精确或非精确）
$\theta_{k+1} \leftarrow \theta_k + \alpha_k p_k$ ▷ 更新参数
$y_k = \nabla\mathcal{L}(\theta_{k+1}) - \nabla\mathcal{L}(\theta_k)$ ▷ 求梯度之差
$\mathcal{H}_{k+1} \leftarrow \mathcal{H}_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{\mathcal{H}_k s_k s_k^T \mathcal{H}_k^T}{s_k^T \mathcal{H}_k s_k}$ ▷ 更新海森矩阵估计
结束 while 循环
返回 θ_k

BFGS算法（算法 2）表明：

$$\theta_{k+1} = \theta_k + \mathcal{H}_k^{-1}\frac{1}{n}\sum_{k=i}^{n}\nabla_\theta\mathcal{L}(x_i;\theta) \tag{2.8}$$

如果 $\mathcal{H}_0 = I$ 是单位矩阵，则第一次迭代等价于最速下降。通过考虑 *Sherman-Morrison* 公式 [35], ，可以直接求得估计海森矩阵的逆，因此

$$\mathcal{H}_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right)\mathcal{H}_k^{-1}\left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k} \tag{2.9}$$

非精确线搜索用于寻找一个能够充分降低 $\mathcal{L}(\theta_k + \alpha p_k)$ 的 α ，而不是寻找精确的最小值，因为这样计算成本更低。从根本上说，梯度下降方法中使用的学习参数在这里被估计的海森矩阵的逆所替代，从而得到更智能的下降方向。这是因为我们对梯度及其变化有近似的了解。如果 $\mathcal{H}_k^{-1}$ 是精确的海森矩阵的逆，且算法成本为二次阶，那么该算法可以通过单步迭代达到最小值。

2.1.4 非精确线搜索

能够“充分降低”目标函数的步长通常根据*Armijo*规则 [36] 和*Wolfe* 条件 [37]来定义。

Armijo规则指出，给定第 k 次迭代的搜索方向 p_k 已找到后，必须满足以下不等式

$$\mathcal{L}(\theta_k + \alpha_k p_k) \leq \mathcal{L}(\theta_k) + c_1\alpha_k p_k^T\nabla\mathcal{L}(\theta_k) \tag{2.10}$$

其中 $0 < c_1 < 1$ （通常较小，例如 $c_1 \approx 10^{-4}$）。也就是说，我们希望更新步长 θ，使其低于从点 θ_k 沿着 p_k 方向回溯线搜索所得的缩减切线。它在以下两者之间取得平衡

1. 不要采取过大的步长，使得 $\mathcal{L}(\theta_{k+1}) \geq \mathcal{L}(\theta_k)$。

2. 不要采取过小的步长，否则收敛所需时间会过长。

由于该算法采用回溯线搜索，它通常会取满足该条件的 α_k 最大值。

关于曲率的 Wolfe 条件指出，必须满足以下不等式

$$|p_k^T \nabla \mathcal{L}(\theta_k + \alpha_k p_k)| \leq c_2 |p_k^T \nabla \mathcal{L}(\theta)| \tag{2.11}$$

其中 $0 < c_1 < c_2 < 1$ （对于 BFGS 这类拟牛顿法，$c_2 \approx 0.9$）。从根本上说，我们预期，如果所采取的步长能够使目标函数最小化，那么该点处的梯度应小于前一个点处的梯度（也就是说，如果我们正趋近于一个最小值，函数应该会变得更加“平坦”）。

2.1.5 验证

人工神经网络训练完成后，可以使用单独的带标签数据集进行测试，以了解它在此前未见过的数据上的表现。这个数据集称为验证数据集 [38]。验证阶段之所以极其重要，原因有很多。

1. 训练数据集可能无法涵盖人工神经网络试图解决的问题的所有方面。使用较小的数据集来测试网络，有助于设计人员发现训练数据中的不足。

2. 它可以揭示过拟合这一机器学习中的常见问题。人工神经网络在训练数据集上的表现可能极佳，但在验证阶段表现不佳。随后可以采用数据增强[5] 和复杂度简化[6], 等方法来避免过拟合。

3. 尽管这对物理信息神经网络而言并非问题，验证阶段也需要确保受保护变量之间的公平性。

[5]数据增强是指设计人员以可衡量的方式扰动原始数据，从而人为地向数据集添加数据，例如镜像翻转图像。[6]复杂度简化是指设计人员简化所用模型，例如移除节点和层。简单来说，网络的节点和层越多，自由度就越高。因此，移除其中一些自由度便可避免过拟合。

第 3 章

用于偏微分方程的神经网络

传统人工神经网络以及求解或近似偏微分方程的常用方法，也会遇到相同的问题——问题定义不明确或数据不足。对于解析解，我们需要在感兴趣区域内，初始条件和边界条件均有对应的闭式方程。要进行精确的有限元分析，通常需要一组足够密集的数据点（通常称为网格）。如果待求解的偏微分方程要求解具有很高的光滑性，网格还需要更加细密，因为各元素在单元边界处必须在 k 阶导数上保持一致。

普通人工神经网络也会存在一些相同的缺点。仅凭数据来近似一个复杂的物理系统（达到较高的容差）可能需要稠密的数据集，而这并不总是可获得的。一个经典问题是求解二维声波方程，其定义为

$$\rho\nabla\cdot\left(\frac{1}{\rho}\nabla u\right)-\frac{1}{v^2}\frac{\partial^2 u}{\partial t^2}=-\rho\frac{\partial^2 f}{\partial t^2} \tag{3.1}$$

其中

- $\rho(x,y)$ 是描述材料在点 (x,y) 处密度的密度函数。
- $u(x,y,t)$ 是波场，即声波产生的压力响应
- $v(x,\ y)=\sqrt{\frac{k}{\rho}}_{(x,\ y)}$ $(x,\ y)$ 是速度，其中 k 为绝热压缩模量[1]
- $f(x,\ y,\ t)$ 是源项，力会注入其中（可能产生声波，例如小型爆炸）。

[1]绝热压缩模量描述了材料被压缩的能力。它描述了材料的体积随压力增加而减小的情况。

该方程可应用于地球建模和地震波测量（第5章中对此进行了探讨）。在没有源项且密度恒定的情况下，该方程可化简为

$$\nabla^2 u - \frac{1}{v^2}\frac{\partial^2 u}{\partial t^2} = 0 \tag{3.2}$$

其中，$\nabla^2 u = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + ... + \frac{\partial^2 u}{\partial x_n^2}$ 是拉普拉斯算子，每个 x_i 都是一个空间变量。这是波动方程最简单的形式，称为规范形式。与数值方法不同，使用物理信息神经网络（PINNs）无需对时间和空间中的区域进行离散化（这一过程通常会引入误差）。因此，PINNs 能够学习时空域之外的行为和输出，并且学习这些行为的速度要快得多。

3.1 PINNs 的损失函数

从根本上说，虽然偏微分方程本身无法直接告诉我们解应当是什么样子，但它确实能指示解应如何变化。以公式3.2为例，偏微分方程告诉我们，解在空间上和时间上都应当二阶可微，并且空间方向上的平均压力之和应等于时间方向上加权后的差值（权重为速度的平方），且这一关系在每个点上都成立。

如果向人工神经网络提供极其密集、覆盖充分且无噪声的数据集，根据通用逼近定理，它最终会收敛到这种行为（假设网络规模和计算能力不受限制）。然而，这种程度的数据并非总能获得。例如，几乎不可能在宽度和深度均达 2 km 的范围内密集测量声波的压力。即使能获得这样的数据集，我们又能以多高的准确度和精度测量这些数据？我们能否考虑噪声的影响？噪声又会如何影响解？物理信息神经网络通过构造特殊的损失函数来弥补数据不足，使网络在最小化已知输出与估计输出之间损失的同时，也遵循偏微分方程所描述函数的预期行为，从而收敛到一个解。

考虑一个人工神经网络 Λ，它用于估计由公式3.1描述的函数 u 的波场行为。假设标准损失函数由公式2.3描述，我们现在将其称为数据损失；另定义一个描述声波的微分算子 $\mathcal{N}$，其形式如下：

$$\mathcal{N}(\Lambda) = \rho\nabla\cdot\left(\frac{1}{\rho}\nabla\Lambda\right) - \frac{1}{v^2}\frac{\partial^2\Lambda}{\partial t^2} + \rho\frac{\partial^2 f}{\partial t^2} \tag{3.3}$$

显然，如果 Λ 是声波方程的精确解，那么 $\mathcal{N}(\Lambda) = 0$。我们可以将其作为约束条件加入损失函数中。也就是说，不

仅期望解 Λ 最小化已知输出$u(x_i,\ y_i,\ t_i)$与估计输出 $\Lambda(x_i,\ y_i,\ t_i;\theta)$之间的差异，还期望该解最小化 $\mathcal{N}(\Lambda)$的数值。因此，物理信息损失的形式为 [57]

$$\mathcal{L}(\theta) = \frac{1-\lambda}{N_d}\sum_{i=1}^{N_d}||u(x_i,y_i,t_i) - \Lambda(x_i,y_i,t_i)||_2^2 + \frac{\lambda}{N_s}\sum_{j=1}^{N_s}||\mathcal{N}(\Lambda(x_j,y_j,t_j))||_2^2 \quad (3.4)$$

我们将第一项称为数据损失，第二项称为物理损失，其中

- $0 \leq \lambda \leq 1$ 是选定的权重参数。数值越大，物理学贡献越大

- N_d 是数据集的大小

- 每个 $u(x_i, y_i, t_i)$ 都是来自大小为 N_d 的数据集中的未知波场函数 u 的已知输出

- 每个 $\Lambda(x_i, y_i, t_i)$ 都是 PINN 在已知数值的点处的输出

- N_s 是整个区域内的采样点数量

- 当前参数下，每个 $\mathcal{N}(\Lambda(x_j,\ y_j,\ t_j))$ 都是 3.3 的一个输出 θ

3.2 神经网络求导

显然，物理信息神经网络高度依赖准确的导数。如第1章所示，为了最小化目标函数，需要根据一部分数据的梯度（或估计梯度）构造雅可比矩阵和海森矩阵，而这些矩阵可能非常庞大。随后利用这些矩阵调整网络参数。如果物理信息神经网络的可调参数数量 θ 为 Λ n，那么 $\mathcal{J} \in \mathbb{R}^n$ 和$\mathcal{H} \in \mathbb{R}^{n\times n}$，且 n 都可能任意大。不仅如此，方程 3.4 中的物理信息损失还需要对输入变量求导，而输入变量的维度也可能很高。因此，显然需要设计一种高效且准确的人工神经网络求导方法。

计算导数时，有四种主要方法可供选择 [58]

- 手动微分：对已知函数（本例中为神经网络 Λ）进行手动求导，直到达到所需的阶数。

- 数值微分：使用有限差分近似。

- 符号微分：使用计算表达式操作库，例如 SymPy。

- 自动微分：依次对 Λ 中的单个算子应用链式法则来计算导数。

3.2.1 微分技术存在的问题

在处理物理信息神经网络时，由于其输入维度可能很高且参数众多，许多这类技术都会带来各自的问题。

手动微分

尽管从理论上说，这将是计算导数最准确的方法，但这项任务极其繁琐。不仅大型人工神经网络包含复杂的表达式，而且需要推导的方程数量也可能多得惊人（可能达到数百万个）。对于 Λ，这意味着必须针对每个输入求 k 阶导数，以构建物理损失；针对 θ 中的每个参数求导，以构建雅可比矩阵；并针对 θ 中每一对可能的参数求导，以构建海森矩阵。当然，人类在计算过程中也容易出错。

问题不仅在于任务规模庞大，而且这种方法完全不具备动态性。每个导数都必须手动编码；如果网络设计发生任何变化，工程师就必须从头开始整个过程。

数值微分

数值微分是一种广泛使用的技术，其通过对函数的泰勒级数展开进行估计来近似计算导数。设有函数 f。通过取泰勒展开 [59]，我们可以以任意精度估计点 x 处的导数数值。

$$f(x+\Delta x) = f(x) + f'(x)\Delta x + \frac{f''(x)\Delta x^2}{2!} + ... + \frac{f^{(n)}(x)\Delta x^n}{n!} + \mathcal{O}(\Delta x^n) \tag{3.5}$$

如果我们取 $n=1$，就能求出一阶导数的近似解，因此

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{3.6}$$

只要步长 Δx 足够小， Δx 项及其后各项就会足够小，可以忽略。不过，我们还需要确保$|\Delta x| > \epsilon$，其中 ϵ 是机器精度[2] , , 以避免舍入误差 [60]。

举例来说，想象加热一块完美的正方形、无限薄的冰淇淋块。考虑定义在 $[0,1] \times [0,1]$ 正方形区域上的二维热方程

[2]机器精度是浮点数（计算算术）相对误差的上界。单精度下为 $\epsilon = 5 \times 10^{-6}$，双精度（大多数现代计算机采用的标准精度）下为 $\epsilon = 5 \times 10^{-15}$，四倍精度（用于专用计算机）下为 $\epsilon = 5 \times 10^{-33}$

并满足以下条件

$$\begin{cases} \phi_t = \alpha(\phi_{xx} + \phi_{yy}) \quad \text{in } \Omega \\ \phi(0, y, t) = 100 \quad (BC_1) \\ \phi(1, y, t) = 25 \quad (BC_2) \\ \phi(x, 0, t) = 200 \quad (BC_3) \\ \phi(x, 1, t) = 0 \quad (BC_4) \\ \phi(x, y, 0) = 50 \quad \text{in } \Omega \quad (IC) \end{cases} \tag{3.7}$$

其中 $\alpha = 1.28 \times 10^{-4}$ 是材料的热扩散率。可以利用泰勒级数得到有限差分近似，因此

$$\begin{aligned} \phi_t &\approx \frac{\phi_{ij}^{(k+1)} - \phi_{ij}^{(k)}}{\Delta t} \\ \phi_{xx} &\approx \frac{\phi_{i-1j}^{(k)} - 2\phi_{ij}^{(k)} + \phi_{i+1j}^{(k)}}{\Delta x^2} \\ \phi_{yy} &\approx \frac{\phi_{ij-1}^{(k)} - 2\phi_{ij}^{(k)} + \phi_{ij+1}^{(k)}}{\Delta y^2} \end{aligned} \tag{3.8}$$

其中，k 是时间步长，i 和 j 分别是空间位置 x 和 y，Δt 是时间步长，Δx 和 Δy 是空间步长。如果取 $h^2 = \Delta x^2 = \Delta y^2$，那么我们可以代入偏微分方程并重新整理，得到

$$\phi_{ij}^{(k+1)} = \left(1 - \frac{4\Delta t\alpha}{h^2}\right)\phi_{ij}^{(k)} + \Delta t\alpha\left(\frac{\phi_{ij-1}^{(k)} + \phi_{i-1j}^{(k)} + \phi_{ij+1}^{(k)} + \phi_{i+1j}^{(k)}}{h^2}\right) \tag{3.9}$$

也就是说，如果我们知道板在某个时间 $t = k$ 的完整状态，就可以利用这些信息来估计板在时间 $t = k + 1$ 的热分布。

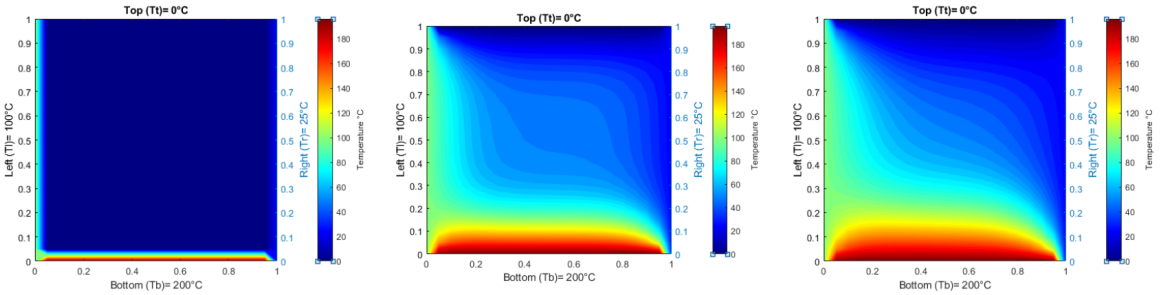


图 3.1：完成示例的热方程有限差分近似（对应 3.7）。从左到右：$t = 0$、$t = 10$、$t = 20$。

显然，有限差分法有其优点。它易于使用和实现，对于较短的时间和较小的区域，计算速度也很快，而且误差容易量化。它也具有相当的灵活性——改变物理信息神经网络的结构，不会影响用于近似梯度的代码或策略。

不过，它也有缺点。其中一个缺点是，我们只能从初始条件 [61] 开始向前求解。如果想知道较大 t 时的解，就需要（准确地）计算截至该点的所有 $\phi(x, y, t)$ 数值。还必须考虑时间和空间上的步长。步长过大，解就不够准确；步长过小，则容易因机器精度而产生舍入误差。对于更复杂的高阶偏微分方程，这个问题会更加严重，例如四阶偏微分方程（有时称为双调和方程）[62] ，这就要求 $h^4 > \epsilon$。这对可用步长构成了实际限制，因此可能导致精度不足。由于每次迭代的精度也取决于上一次迭代的精度，误差很容易向前传播。四阶偏微分方程并不少见。其中一个特别著名的例子出现在结构力学和工程领域，即板弯曲问题

$$\nabla^4 w = \frac{p}{D} \tag{3.10}$$

其中， p 表示板上的载荷分布， D 表示杨氏模量[3]。也存在阶数更高的偏微分方程（具有实际物理应用），但这类方程要少得多。Kaup–Kupershmidt 方程[63], 是一种五阶偏微分方程，其应用与更著名的科尔特韦格–德弗里斯（KdV）方程 [64]类似，后者用于浅水波建模。

机器学习依赖准确的梯度来优化人工神经网络中（有时多达数百万个）参数，因此在训练阶段引入舍入误差和截断误差，可能会严重影响人工神经网络收敛到合适解的能力；尤其是使用随机方法时，梯度本身就只是对空间的估计值。步长选择不当也可能导致数值不稳定性。此外，对于物理信息神经网络，损失函数将偏微分方程纳入优化过程，因此高阶偏微分方程也会遇到与有限差分法相同的问题。

符号微分

符号微分会获取由函数描述的表达式，并应用微分规则（乘积法则、链式法则等），以返回一个新函数 f'，它是函数 f 的微分结果。事实上，这与手动微分非常相似，只不过由计算机执行计算并返回一个新表达式。这是

[3]杨氏模量以托马斯·杨命名，是材料的拉伸应力强度与材料应变强度之比。

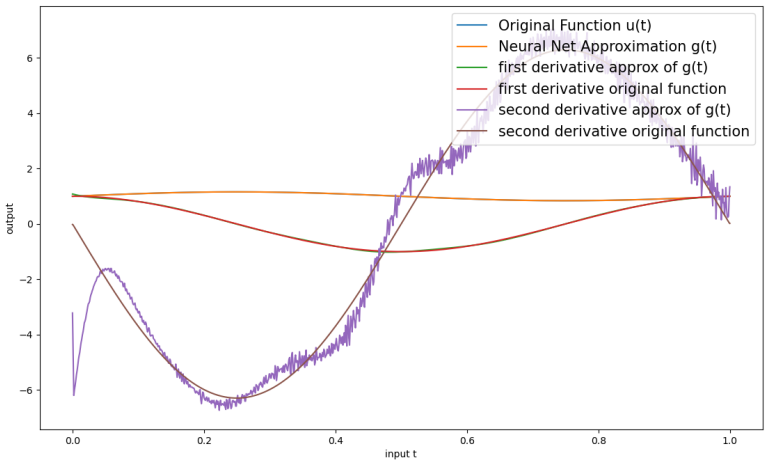


图 3.2：人工神经网络近似对一个简单常微分方程的解进行估计。近似解极其准确，一阶导数也是如此。尽管如此，使用有限差分法估计二阶导数，仍然可以看出，与实际的二阶导数相比，我们会损失多少光滑性。

之所以可行，是因为在求导时，实际上只有 8 条规则需要遵循——因此，难点在于正确应用这些规则。

符号微分面临着困扰前两种方法的许多问题。计算机负责繁重的表达式计算工作，从而解决了应用手动微分时的主要问题。此外，所得的解是精确的，因此我们可以避免根据泰勒级数展开进行估计所带来的数值误差。

然而，将其应用于计算函数时也会遇到问题，尤其是应用于人工神经网络时。首先，它无法考虑条件计算逻辑（例如 if、while 和 for）。其次，人工神经网络需要用闭式表达，这在技术上虽可行，却增添了另一项挑战。第三，符号微分会导致表达式极度膨胀 [58]，（附录 A.1.1 中有一个例子）。某些求导规则（如乘积法则）会自然地使方程中的项数增加，因此计算导数可能会非常繁琐。

3.2.2 自动微分

自动微分（AD）解决了上述许多挑战，因为它能够以机器精度计算某一点处的导数。它通过利用计算数学的两个方面来实现这一点——对偶数 和计算图表示 [65]。从根本上说，所有函数都由可微分的简单运算组成

以及用于存储函数在不同点处信息的中间变量。自动微分利用这一事实，并运用链式法则，将较小子函数的导数结合起来，直接求出导数的数值，而不是尝试计算闭式表达式，或利用邻近点进行估计。

对偶数

对偶数是在浮点运算中表示数值的一种特殊方式[66], ，可用于在计算主要输出的同时计算函数的导数。设 z 为一个浮点数， ϵ 为无穷小量。那么我们可以说 $z = a + b\epsilon$ ，它可以用矩阵形式表示为

$$a = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \quad b\epsilon = \begin{bmatrix} 0 & b \\ 0 & 0 \end{bmatrix} \tag{3.11}$$

因此

$$\epsilon = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \implies \epsilon^2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.12}$$

现在，取一个次数为 N 的一般多项式 P，它可以简洁地表示为

$$P(x) = a_0 + \sum_{n=1}^{N} a_n x^n \tag{3.13}$$

由此可得

$$\begin{aligned} P(x+\epsilon) &= a_0 + \sum_{n=1}^{N} a_n (x+\epsilon)^n \\ &= a_0 + a_1(x+\epsilon) + a_2(x+\epsilon)^2 + ... + a_N(x+\epsilon)^N \\ &= a_0 + a_1(x+\epsilon) + a_2(x^2 + 2\epsilon x + \epsilon^2) \\ &+ a_3(x^3 + 3\epsilon x^2 + 3\epsilon^2 x + \epsilon^3) + ... \end{aligned} \tag{3.14}$$

然而，由于 $\epsilon^2 = 0$，所有满足 $k \geq 2$ 的项 ϵ^k 都可以去掉，因此

$$\begin{aligned} P(x+\epsilon) &= a_0 + a_1(x+\epsilon) + a_2(x^2 + 2\epsilon x) \\ &+ a_3(x^3 + 3\epsilon x^2) + ... + a_N(x^N + N\epsilon x^{N-1}) \\ &= a_0 + \sum_{n=1}^{N} a_n x^n + \epsilon \sum_{n=1}^{N} a_n n x^{(n-1)} \\ &= P(x) + \epsilon \frac{\partial P(x)}{\partial x} \end{aligned} \tag{3.15}$$

因此，通过将给定数值 x 以对偶数形式代入并计算 $P(x+\epsilon)$，我们一次计算就得到了 x 点处的 P 数值以及 x 点处的导数，且达到

机器精度。一般而言，如果函数 f 在点 x 处可微，则 n

$$f(x+\epsilon) = f(x) + \epsilon\frac{\partial f(x)}{\partial x} \tag{3.16}$$

前提是 ϵ^2 在计算上等于 0。

计算图

计算图只是一种表示函数的方式，可以让人轻松理解数据 [67]的流动。它与图 1.2 中人工神经网络的表示方式非常相似，只是更加通用一些（因为它适用于所有类型的函数，而不仅仅是人工神经网络），也更加详细，因为它还展示了各个步骤之间涉及的运算。

举个简单的例子，设想一个函数 $f : \mathbb{R}^2 \to \mathbb{R}$。如果 $x \in \mathbb{R}^2$，并且

$$f(x) = \left(\frac{x_1}{x_2} + \cos(x_1)\right)\left(\frac{x_1}{x_2} + e^{x_2}\right) \tag{3.17}$$

那么它的计算图表示可以如图 3.3 所示。

这种表示方式有三方面的用处。首先，它以一种较易理解的方式展示了给定函数中的变量是如何参与运算的。其次，它使算法能够利用重复出现的运算和变量。举例来说，除法 x_1/x_2 在公式3.17中出现了两次，而图中利用了这一信息，只执行了一次该运算。第三，这种表示方式使我们能够系统地对中间变量应用链式法则，从而高效地计算函数相对于输入的偏导数，而且整个过程只需一次遍历即可完成。

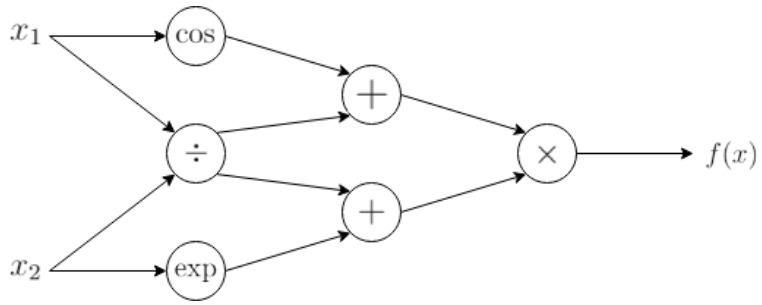


图 3.3：方程 3.17 的计算图表示，展示了如何对 2 个变量进行组合运算，从而生成任意输出

计算导数

如前所述，自动微分不会求出相对于任意变量的导数表达式，而是求出相对于任意变量的导数在特定点处的数值。它通过利用链式法则来实现这一点，而这可以通过两种方式完成——正向模式 自动微分或反向模式 自动微分。

当我们要微分的函数 f 映射到的输出数量多于输入数量时，前向自动微分更高效；反之，反向模式自动微分更高效。以公式 3.17 中定义的示例为例，不过这次我们要记录中间变量的数值。

在计算每个变量的数值（通常称为原始量）时，我们也可以计算每个变量相对于某个输入的梯度（通常称为切向量）。暂且假设输入向量为 $x = [1, 2]^T$，并且我们需要求出此点处 f 的数值，以及相对于 x_1的梯度。将 $\dot{v}_i$ 定义为相对于 x_1求导的任意变量后，我们便可以利用对偶数的性质进行前向传播，计算每个变量的原始量和切向量。

图 3.4 和表 3.1 显示了 $\frac{\partial f}{x_1} = \dot{w} = =2.173732$。我们可以通过在该点计算导数的表达式来验证这一点（这里可以这样做，因为 f 很简单），因此

$$\frac{\partial f}{\partial x_1} = \left(\frac{x_1}{x_2} + \mathrm{e}^{x_2}\right)\left(\frac{1}{x_2} - \sin\left(x_1\right)\right) + \frac{\cos\left(x_1\right) + \frac{x_1}{x_2}}{x_2} \tag{3.18}$$

因此，$\frac{\partial f}{\partial x_1}\Big|_{x_1=1,\ x_2=2}$ 为 -2.173732，正如预期（事实上，结果精确到小数点后 14 位，符合我们对机器精度的预期）。使用步长为 $\Delta x_1 = 1 \times 10^{-5}$ 的有限差分，得到 $\frac{\partial f}{\partial x_1}\Big|_{x_1=1,\ x_2=2} \approx -2.1738$，结果仅精确到小数点后 3 位，因此即使对于这个非常简单的函数，我们也会损失大量精度。

因此，原始量和切向量可以并行计算。使用前向模式自动微分，一次计算即可生成雅可比矩阵的一整列；此外，还能利用中间数值和重复运算，从而减少所需的计算量

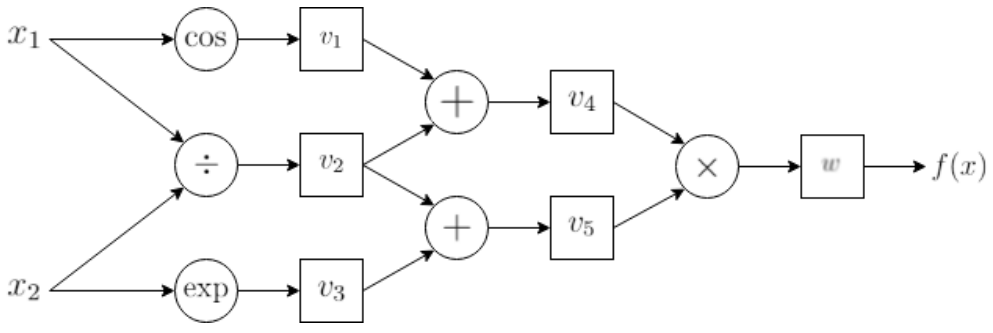


图 3.4：方程 3.17 的计算图表示，此处存储数值

原始量		切向量	
表达式	数值（保留 6 位小数）	表达式	数值（保留 6 位小数）
$x_1 = x_1$	1	$\dot{x_1} = 1$	1
$x_2 = x_2$	2	$\dot{x_2} = 0$	0
$v_1 =$余弦(x_1)	0.540302	$\dot{v_1} = \dot{x_1}$正弦(x_1)	-0.841471
$v_2 = \frac{x_1}{x_2}$	0.5	$\dot{v_2} = x_2^{-1}$	0.5
$v_3 = e^{x_2}$	7.389056	$\dot{v_3} = 0$	0
$v_4 = v_1 + v_2$	1.040302	$\dot{v_4} = \dot{v_1} + \dot{v_2}$	-0.341471
$v_5 = v_2 + v_3$	7.889056	$\dot{v_5} = \dot{v_2} + \dot{v_3}$	0.5
$w = v_4 v_5$	8.207001	$\dot{w} = \dot{v_4} v_5 + v_4 \dot{v_5}$	-2.173732

表 3.1：此处列出了图 3.4 中计算图的中间变量的数值。利用链式法则，一次前向传播即可计算函数在给定点处的值及其梯度。计算机并不会真正“知道”切向量的表达式——这些数值是在使用对偶数时作为副产品计算出来的。随后利用链式法则计算导数。

，提高效率。对于 $m > n$，这是计算梯度的高效方法。不过，如果$n > m$，我们可以改用反向模式自动微分，逐行构造雅可比矩阵。按列构造雅可比矩阵，意味着每次计算都会求出所有输出相对于一个输入的数值；按行构造则会求出单个输出相对于各个输入的数值。运行前向模式自动微分的成本为 $\mathcal{O}(n)$，而反向模式的成本为 $\mathcal{O}(m)$。其他求导模式要么会引入误差，要么是 $\mathcal{O}(nm)$ [68]。

物理学中的许多函数具有高维输入和低维输出，这意味着反向模式自动微分非常适合用于计算导数。这一点在机器学习中尤为明显，因为参数规模通常显著大于输出规模。

3.3 通用逼近器

虽然物理信息神经网络的主要驱动力在于损失函数的定义方式，但设计网络时还有其他因素需要考虑。首先，我们应该思考还能将哪些信息直接编码到物理信息神经网络中，以加快收敛速度。

这类信息中有两项：边界条件和初始条件。以对 2×3矩形区域内的二维振动膜进行建模为例。将边缘固定（狄利克雷 边界条件[4]），并定义

[4]狄利克雷边界条件以彼得·古斯塔夫·勒让·狄利克雷命名，指函数在区域边界上取特定的固定解。

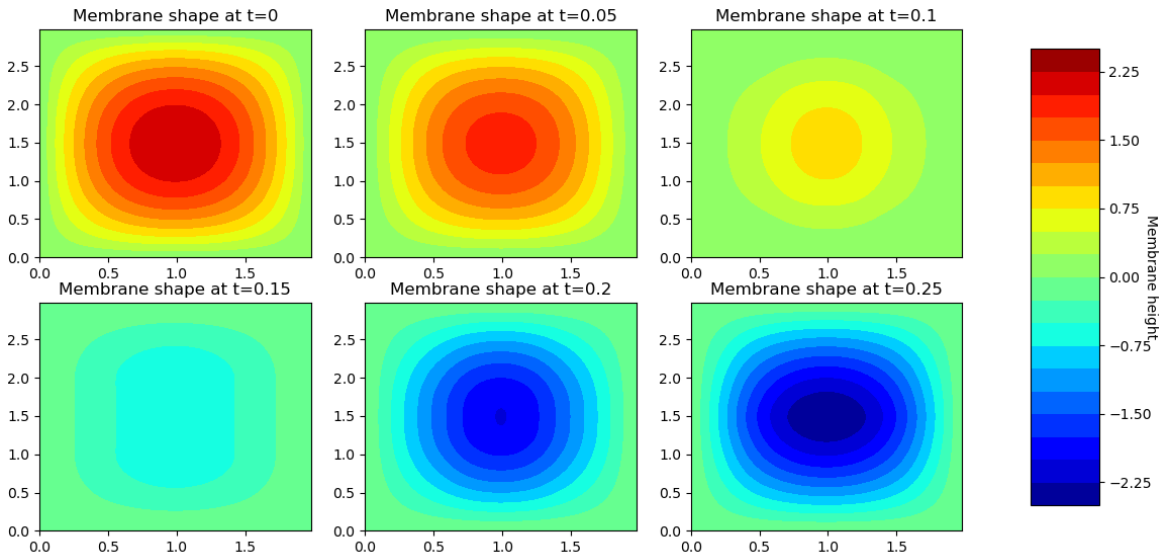


图 3.5：如 3.19 所述，具有狄利克雷边界条件且无阻尼的简单振动膜的级数解

膜在区域内的初始形状和速度，使得

$$\begin{cases} u_{tt} = 6^2(u_{xx} + u_{yy}) \quad \text{in } \Omega \\ u = 0 \quad \text{on } \partial\Omega \quad (BC) \\ u = xy(2-x)(3-y) \quad \text{for } t = 0 \quad (IC_1) \\ u_t = 0 \quad \text{for } t = 0 \quad (IC_2) \end{cases} \tag{3.19}$$

因此区域为 $[0,2] \times [0,3]$。这个偏微分方程有一个级数解

$$u(x,y,t) = \frac{576}{\pi^2} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \left(\frac{(1+(-1)^{i+1})(1+(-1)^{j+1})}{i^3 j^3} \sin\left(\frac{i\pi x}{2}\right) \right. \\ \left. \sin\left(\frac{j\pi y}{3}\right) \cos(\pi\sqrt{9i^2+4j^2}t) \right) \tag{3.20}$$

令 U 为通用逼近器，并令 $\Lambda(x,y,t)$ 为未训练的物理信息神经网络。在这种情况下，损失函数中的微分算子仅为

$$\mathcal{N}(\Lambda) = \Lambda_{tt} - 6^2(\Lambda_{xx} + \Lambda_{yy}) \tag{3.21}$$

因此，损失函数的定义与方程 3.4 中完全一致，只是采用了上文所述的新微分算子。最初的想法是，简单地创建通用逼近器 $U(x,\ y,\ t)$，使其仅等于 PINN [69], so

$$U(x,y,t) = \Lambda(x,y,t) \tag{3.22}$$

然而，这意味着 Λ 必须先学习初始条件和边界条件，然后才能将时间用于学习如何映射函数的其余部分。显然，既然这里已经知道这些信息，更明智的做法是预先将这些信息提供给 PINN [70]。为了编码初始条件，我们可以改为定义

$$\begin{aligned} U(x,y,t) &= t\Lambda(x,y,t) + u(x,y,0) \\ &= t\Lambda(x,y,t) + xy(2-x)(3-y) \end{aligned} \tag{3.23}$$

损失函数为

$$\mathcal{L}(\theta) = (1-\lambda)\frac{1}{N_d}\sum_{i=1}^{N_d}||u(x_i,y_i,t_i) - U(x_i,y_i,t_i)||_2^2 + \lambda\frac{1}{N_s}\sum_{j=1}^{N_s}||\mathcal{N}(U(x_j,y_j,t_j))||_2^2 \tag{3.24}$$

现在，只要 $t = 0$ ，物理信息神经网络就会自动完美满足初始条件，无需对这种行为进行任何近似。我们也可以通过进一步扩展通用逼近器，对边界条件进行完全相同的处理，使得

$$\begin{aligned} U(x,y,t) &= u(\partial\Omega,t)t\Lambda(x,y,t) + u(x,y,0) \\ &= xy(2-x)(3-y)t\Lambda(x,y,t) + xy(2-x)(3-y) \end{aligned} \tag{3.25}$$

由于 $f(x,\ y) = y(2-x)(3-y) = 0$ 对于 $x = 0$、$x = 2$、$y = 0$ 或 $y = 3$（感兴趣区域的边缘部分）成立，因此这满足了边界条件。这一点小小的处理意味着，在训练阶段开始之前，通用逼近器就能在所有时间 $t \geq 0$ 上完美地模拟边界条件，并在区域内所有空间位置上完美地模拟初始条件。

需要注意的是，这并不总是可行的。我们可能根本不了解边界条件/初始条件，数据可能存在噪声或较为稀疏，或者边界可能是开放的或定义不明确的。不过，这确实表明，存在一些训练阶段之外的方法，可以确保物理信息神经网络收敛到真实解。

第4章

简单的物理信息神经网络示例

只要问题定义明确，前面各章介绍的所有材料都可以用于为几乎任何问题构建物理信息神经网络。对于定义不明确的问题，物理信息神经网络仍会收敛到它能够找到的最接近的解，而它最终收敛到哪个解，通常取决于初始参数、它能够获取的数据以及网络的结构。

4.1 通用框架

本节采用了 M. Raissi 等人（2017） [49]构建的框架。该论文指出，利用物理信息神经网络，可以高精度地求解定义良好的线性和非线性偏微分方程问题。不仅如此，后续的第二篇论文还表明，只要在感兴趣区域内提供足够的数据，就可以重构偏微分方程中的函数和系数 [50]。这两项发现为下一章讨论的进一步工作奠定了基础。

代码（详见 A.4.2，其中对用法有更多说明）在 Python 中使用 TensorFlow 软件包和 L-BFGS优化算法来求解偏微分方程的近似解。为了得到有意义的结果，程序需要用户提供以下 5 项内容：

1. 空间维度和时间维度上的区域大小，即 $a \leq x \leq b$和 $0 \leq t \leq T$。2. 问题在 $t = 0$ 时空间区域上的初始条件。这既可以用函数表示，也可以用离散数据点表示。3. 给定的边界条件。同样，可以用适用于所有 $0 \leq t \leq T$ 的函数表示，也可以用离散点表示。4. 网络结构（输入规模、输出规模、隐藏层数量以及每层的节点数）。

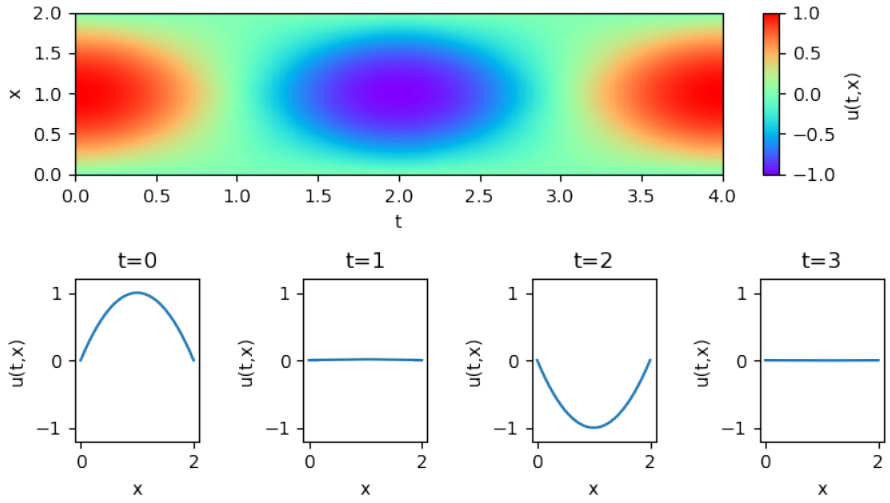


图 4.1：4.1 节所描述的波的行为。由于不存在阻尼，该波将始终如此振荡，其周期为 4

5. 用于创建自定义损失函数的控制方程

解的精度、最大迭代次数、数据样本量等其他参数也可以更改，不过系统提供了默认选项。这是因为获取解所需的时间与物理信息神经网络提供的精度之间需要权衡。

4.2 一维波动方程

4.2.1 解析解

为了展示一个简单直观的示例，考虑具有以下条件的一维波动方程

$$\begin{cases} u_{tt} = u_{xx} & \text{for } 0 \le t \le 4, \quad 0 \le x \le 2 \\ u(x,0) = x(2-x) & \text{IC} \\ u_t(x,0) = 0 & \text{IC} \\ u(0,t) = u(2,t) = 0 & \text{Boundary Condition} \end{cases} \tag{4.1}$$

其解析解如下（证明见附录 A.1.2）

$$u(x,t) = \sum_{n=1}^{\infty} -\frac{8\pi n \sin(\pi n) + 16\cos(\pi n) - 16}{\pi^3 n^3} cos\left(\frac{n\pi t}{2}\right) \sin\left(\frac{n\pi x}{2}\right) \tag{4.2}$$

4.2.2 神经网络

物理信息神经网络要逼近该解，完全不需要任何额外信息。事实上，即使提供更少的信息，我们也可以逼近一个接近的解，因为初始条件和边界条件数据集可能是离散的。在这种情况下，边界条件和初始条件都经过了离散化，因此网络并不了解初始条件或边界条件在所有时间上的行为，只知道特定点处的情况。

对于一维波动问题，PINN $\Lambda(\theta)$ 必须最小化

$$\mathcal{L}(\theta) = \sum_{i=1}^{N_i} \Big(u(x_i, 0) - \Lambda(x_i, 0; \theta) \Big) + \sum_{j=1}^{N_b} \Big(u(x_b, t_j) - \Lambda(x_j, t; \theta) \Big) + \sum_{k=1}^{N_p} \Big(\mathcal{N}(\Lambda(x_k, t_k; \theta)) \Big) \tag{4.3}$$

其中

- 第一项求和是初始条件的损失项，其中 $N_i = 1000$ 是沿初始条件选取的等距点
- 第二项求和是边界的损失项，其中 $N_j = 1000$ 是沿边界随机选取的点，因此 $t_j \in [0, T]$ 和 $x_b \in \{0, 2\}$
- 第三项是物理损失，其中 $N_p = 10000$ 是在整个区域内随机选取的点，且 $\mathcal{N}(\Lambda) = \Lambda_{tt} - c^2 \Lambda xx$ （其中 c=1 是该问题的波速）

导数通过对偶数和 Tensorflow 内置的$GradientTape$ 函数进行自动微分计算，该函数允许程序监视网络相对于输入的运算。这些函数可以嵌套调用，以计算高阶导数。

4.3 误差分析

图4.2中物理信息神经网络给出的近似解显然拟合得相当好。总体行为正确，$t = 0$、1、2、3处的解快照也与解析解非常相似。由于本例中解析解已知，因此可以进行较为详细的误差分析。

对这个非常简单的问题进行误差分析至关重要，因为这有助于在为更复杂的波相关问题创建物理信息神经网络时，做出更明智、更合理的设计选择。了解误差出现的位置、大小以及产生原因十分重要。

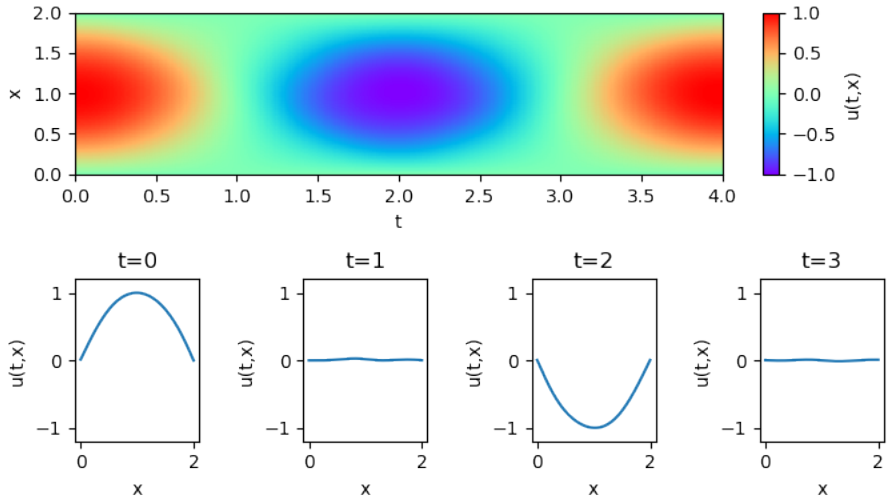


图4.2：物理信息神经网络对波的近似表现。对于上述示例，网络结构为 $[64, 32, 16, 8],$，即包含 4 个隐藏层，层级逐渐递减。

4.3.1 误差测量

要量化解与近似之间的误差可能颇具挑战。不过，在估计解析解与物理信息神经网络解之间的误差时，有三种可能的思路可供借鉴。

- 能量误差。
- 均方误差（MSE）。
- 相对 L_2 范数误差。

通常，我们只能求出误差的上界，因此只能了解最坏情况。有限元分析通过研究解与近似在能量范数意义下的差异来估计误差 [51] [52]。

$$\begin{aligned}\int_\Omega (\nabla u - \nabla u_h) \cdot (\nabla u - \nabla u_h) &= ||u - u_h||_E^2 \\ &= ||u||_E^2 - ||u_h||_E^2\end{aligned} \tag{4.4}$$

其中， u 是精确解， u_h 是近似解，而$\int_\Omega$是区域积分。作为替代，有限元分析使用参考解来确定随着网格细化，解之间的能量范数如何变化，并检验其是否收敛。

$$||e_{ref}||_E^2 = ||u_{ref}||_E^2 - ||u_h||_E^2 \tag{4.5}$$

其中 u_{ref} 是细网格上的参考解， u_h 是较粗网格上的解。这个参考误差可以通过考虑可能出现的最大误差来界定，而最大误差

出现在最长边所在的单元上；也可以通过在细网格上运行计算来明确求得。值得记住这种参考方法，因为我们并非总能获得真实解 u。不过，在本例中，我们确实知道真实解，因此可以直接使用真实的能量范数差。于是，对于这个特定问题，能量范数误差为

$$e_{energy} = \int_0^2 \int_0^4 (\nabla u - \nabla \Lambda) \cdot (\nabla u - \nabla \Lambda) dt dx \tag{4.6}$$

由于我们处理的是区域内连续的两个函数，因此能量范数误差是一个很好的近似度量，因为其他度量要求我们对区域进行离散化。

均方误差 定义为

$$e_{MSE} = \frac{1}{N_e} \sum_{i=1}^{N_e} \Big(u(x_i, t_i) - \Lambda(x_i, t_i) \Big)^2 \tag{4.7}$$

其中， N_e 是数据样本的数量，数据对（x_i， t_i）是采样得到的数据点。从根本上说，它衡量的是区域内有限数量点上的平均平方误差。这些点可以随机选取，但在这里，我们在 x 和 t中取了由 200×400 个等距点组成的定义网格，从而得到 $N_e = 80000$。

最后一种可能的方法用于线性代数，即相对 L_2 范数误差。同样地，我们从解空间和近似空间中获取离散的数据点样本，并分别将它们存储为向量 x 和 b （在本例中再次为80000个）。然后，我们求解与解向量的欧几里得距离相比，解向量与近似向量之差的欧几里得距离。

$$e_L = \frac{\sqrt{\sum_{i=1}^{N_e} (x_i - b_i)^2}}{\sqrt{\sum_{i=1}^{N_e} x_i^2}} = \frac{||x - b||_2}{||x||_2} \tag{4.8}$$

4.3.2 架构设计

在设计任何类型的人工神经网络时，选择隐藏层的数量以及每层的节点数都是重要的一步。令人惊讶的是，围绕某些架构为何以及如何优于其他架构，几乎没有多少可靠的理论。2013年，当被问及“我应该如何知道要使用多少个隐藏层”时，蒙特利尔学习算法研究所所长 Yoshua Bengio 表示“非常简单。不断增加层数，直到测试误差不再改善为止。”

一种思路是研究数据集，看看需要多少条首尾相连的直线，才能将数据充分划分为已定义的组。每条直线之间的连接数就是隐藏层 1 中的节点数，而每组彼此相连的直线之间的连接数就是该层的节点数

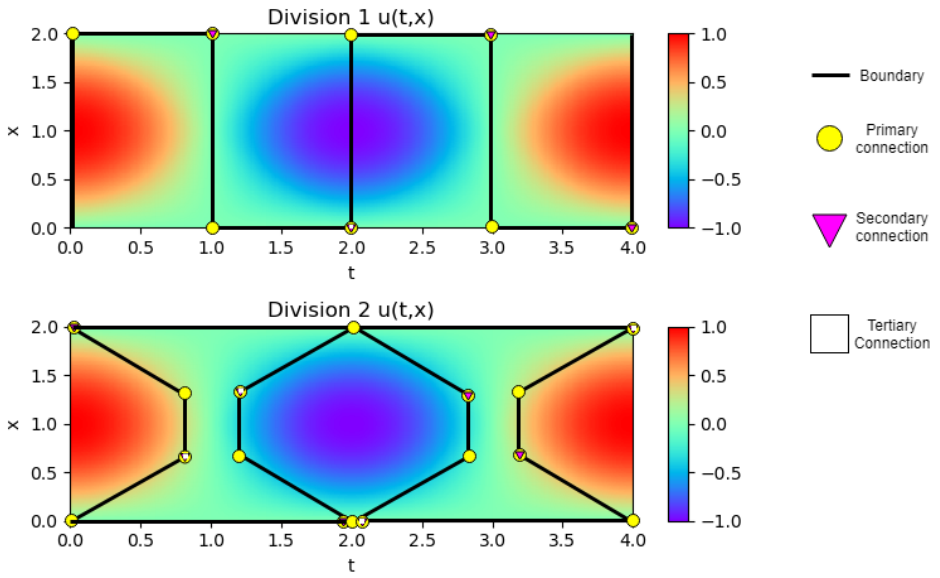


图 4.3：波的行为的两种可能的简单划分。划分 1 将数据划分为导数性质发生变化的区段；划分 2 则根据数据值与 -1、0 或 1 的接近程度，将数据分为不同子集。划分 1 对应一个 8-4-2 网络，而划分 2 对应一个 16-8-4-2 网络。

2，依此类推，直到所有线条连接起来 [53]。

大多数理论采用逆向思路。也就是说，我们先创建一个肯定能奏效的大型人工神经网络，然后通过一个称为 剪枝 [54], 的过程移除层。一条实用的经验法则是，在训练后分析权重和偏置。权重约为 0（或相较于其他权重实际上为 0）时，通常可以将其移除，因为这表明它们对输出的影响很小。另一种方法是不断添加层，直到误差不再减小，正如前文所述。此时，再移除一层，使函数行为不那么针对训练数据。两种技术也都可以防止过拟合。

这里应用了几种不同的物理信息神经网络结构，主要研究层深度如何影响物理信息神经网络的收敛能力。

观察表 4.1 可知，层深度与节点数量之间显然需要取得平衡。尽管参数数量最多的网络确实得到了最准确的近似，但参数越多并不总意味着准确度越高。其他简单得多的架构也具有相当不错的准确度，而训练阶段需要权衡的因素非常多。训练 $[64, 64]$网络耗时超过一小时，而训练较小的网络（例如 $[8, 4, 2]$网络）只需几分钟。不过，最重要的一点是，即使是简单的网络，也能捕捉振动波的总体行为。

图 4.4 展示了参数最少的最简单网络的最终解。虽然这里显示的误差是所有网络中最大的，但它

结构	参数	e_{energy}	e_{MSE}	e_L
128	513	8.858×10^{-5}	1.956×10^{-4}	2.7223×10^{-2}
$64-64$	4,417	3.622×10^{-6}	5.509×10^{-5}	1.444×10^{-2}
$20-20-20-20$	1,341	3.522×10^{-4}	2.721×10^{-4}	3.204×10^{-2}
$32-16-16-32$	1,473	4.475×10^{-5}	1.711×10^{-4}	2.541×10^{-2}
$64-32-16-8$	2,945	1.1815×10^{-5}	7.648×10^{-5}	1.699×10^{-2}
$8-4-2$	73	3.10×10^{-3}	5.954×10^{-4}	4.742×10^{-2}
$16-8-4-2$	233	1.11×10^{-3}	5.262×10^{-4}	4.458×10^{-2}

表 4.1：不同神经网络结构在各种误差范围内的误差表。即使可训练参数的数量很少，大多数架构也能对解给出合理的估计值。然而，在英特尔四核 i7 上训练一个含有 4417 个参数的系统耗时超过一小时（30000 次迭代），而含有 73 个参数的系统则不到一分钟。这就是为什么找到足够小的网络至关重要。

从图中可以清楚地看出，该方法捕捉到了总体行为。由于训练该网络所需的时间极短，因此可以将其用作快速估计值，帮助设计人员了解如何构建更智能的网络。使用图形处理器（GPU）而非 CPU，可大幅缩短收敛所需的绝对时间，但遗憾的是，此处无法采用这种方式。这里使用 L-BFGS 寻找最小值，结果为 $m = 50$。

4.3.3 解收敛

了解整个近似解的总误差只是问题的一半。更值得探究的是这些误差出现在哪里，以及出现误差的原因。举例来说，我们来看看 $[64, 32, 16, 8]$网络给出的估计值。

从表 4.1 和图4.5来看，该解的拟合效果相当不错。相对误差有时较大，但这是有充分原因的。相对误差的计算方式为

$$e_{rel} = \frac{\Lambda(x,t)}{u(x,t)} - 1 \tag{4.9}$$

当 $u(x,\ t) \approx 0$ 时，由于用于计算它们的方程的性质，会出现较大的相对误差。在其他地方，我们也可以看到，当 $|u(x,\ t)| > 0$ 时，相对误差非常小，但绝对误差中明显存在交叉模式。对于这个网络，我们跟踪了收敛情况。具体做法是每隔 50 次迭代对近似空间进行一次快照，然后通过 L_2 误差将其与真实解进行比较。

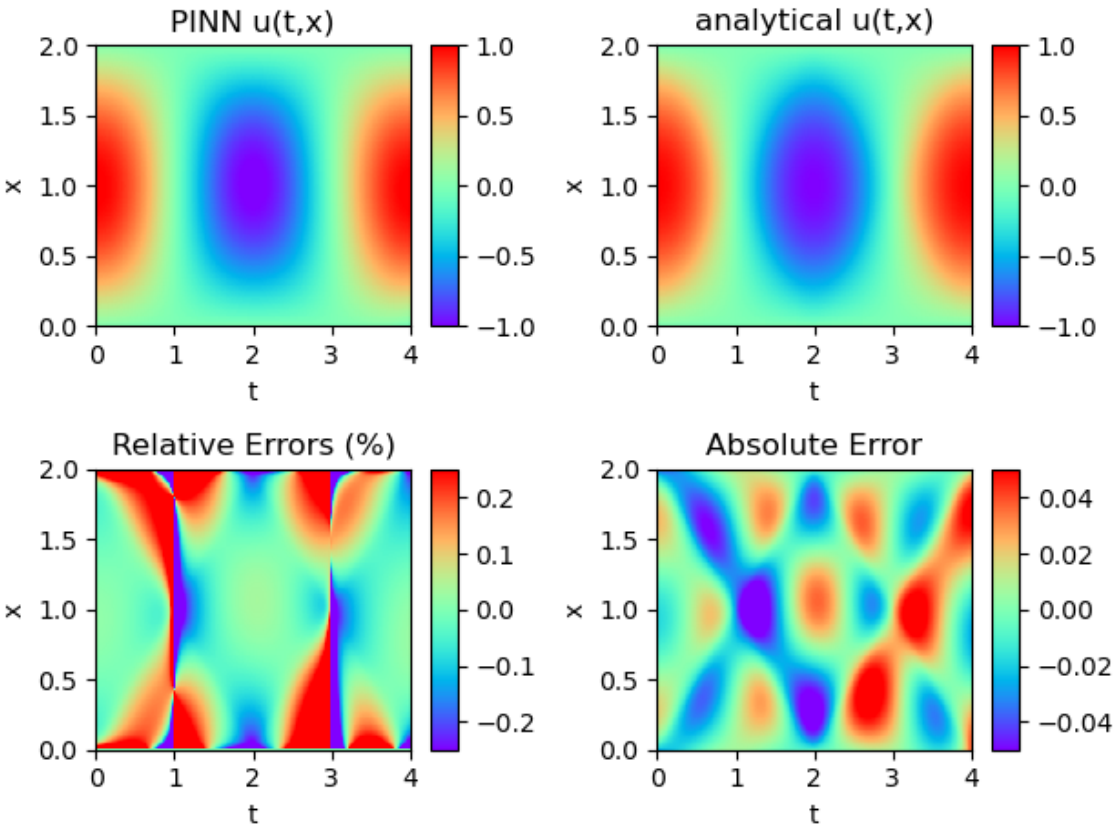


图 4.4：真实解与 8-4-2 网络之间的误差；这是所测试的最简单网络。

该解近似以 $\mathcal{O}(N^{\frac{1}{2}})$ 的速率收敛，因此

$$e_{conv} = \gamma N^{-\frac{1}{2}} \tag{4.10}$$

其中，N 是执行的迭代次数。在此情况下，$\gamma \approx \frac{5}{2}$。有限元分析 [55], 的收敛速率也相同，其中 N 表示网格中的节点数（也称为自由度）。然而，更值得关注的是这种收敛是如何发生的，以及较大误差出现在哪里。

从图4.4和4.5中，似乎可以得出两个主要结论。

1. 误差似乎从初始条件处向外“传播”，初始条件附近区域的绝对误差较小。

2. 误差呈现出某种结构或模式。这种结构或模式在不同的网络架构之间似乎具有一致性（图 4.4 和图4.5 就是很好的例子）——参数较少的网络，其误差只不过更加明显。

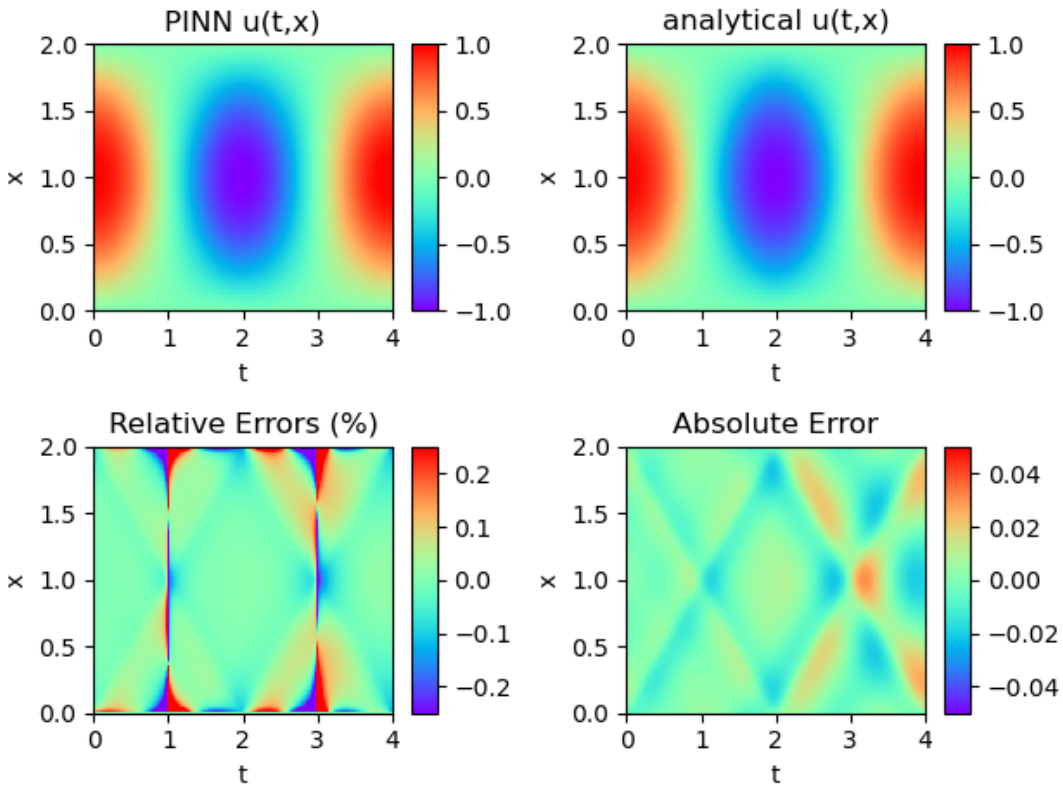


图4.5：真实解与 64-32-16-8 网络之间的误差

4.4 改进学习过程

上一节描述了物理信息神经网络如何收敛到真实解 $u(x,\ t)$。了解这一点后，我们就可以推断如何改进学习过程的性能，以及为何要这样做；随后，这些改进可以应用于规模更大、复杂度更高的问题（将在后续章节中探讨）。

4.4.1 误差传播

关于误差，有一点值得注意：对于一维波动问题的近似解，随着我们远离初始条件，误差似乎会增大，而图 4.7 证实了这一点。

对于这个问题，收集到的所有数据都来自两个地方

1. 问题的边界，其中 $u(0,\ t) = u(2,\ t) = 0$

2. 问题的初始条件，其中 $u(x,\ 0) = x(2 - x)$

因此，我们可以推断，要在 $(x = x_1,\ t = t_1)$ 处获得较低的误差，网络必须先在 $(x = x_n,\ t < t_1)$ 处有良好的近似，其中 x_n 是 x_1 周围邻域中的一个点。回想起网络通过学习

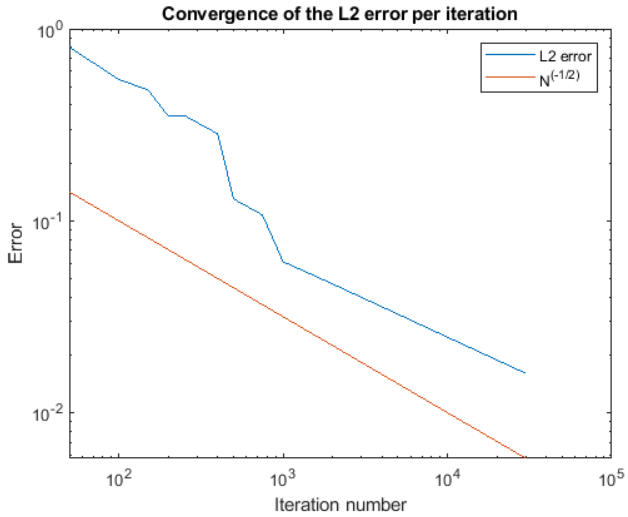


图4.6：随着最小化的进行， L_2 误差的收敛情况

二阶导数来学习行为，这一结论便合情合理。换句话说，要优化空间和时间中的单个点，就需要对空间和时间中的周围点有良好的估计值，而最佳的初始估计总是位于区域中数据密集的点附近。

如果我们不从初始条件中获取数据，而是在$t = 2$处获取相同数量的数据（其中 $u = x（x - 2）$），那么我们应该预期近似会从区域中心开始向外收敛。

图4.8表明，预测似乎是正确的。我们可以清楚地看到，优化必须先在给定数据附近找到一个良好的解，然后才能找到更接近 $t = 0$ 和 $t = 4$的解。除此之外，误差的模式似乎仍与之前的近似（其数据严格取自 $t = 0$）呈现相同的形状。稍后将对此进行讨论。

大多数物理信息神经网络的最小化算法（包括本节所用的算法）都会在每次迭代时从区域中随机抽取样本，以最小化整个区域上的物理损失项。然而，误差传播表明，如果 $t = t_0$ 处的行为近似得很差，那么试图学习 $t = t_0 + t_1$ 处满足 $|t_1| \gg 0$ 的行为就有些徒劳；而且， $u（x，t）$ 在 $t = t_0$处（大部分）已知。这一逻辑引出了一种潜在理论：如何用更少的迭代次数，或每次迭代使用更少的数据点来收敛到一个解，从而提高计算速度。

定理 1 设 Y 为偏微分方程所定义的区域。设 $X^{(0)} = X_1^{(0)} \cup X_2^{(0)} \cup .. \cup X_n^{(0)} \subseteq Y$，其中 $X_k^{(0)}$ 是已知数据密集的区域部分，并以如下方式分区： $X_i^{(0)} \cap X_j^{(0)} = \varnothing$，对于 $i \neq j$。那么，在前 m 次迭代中，或直到

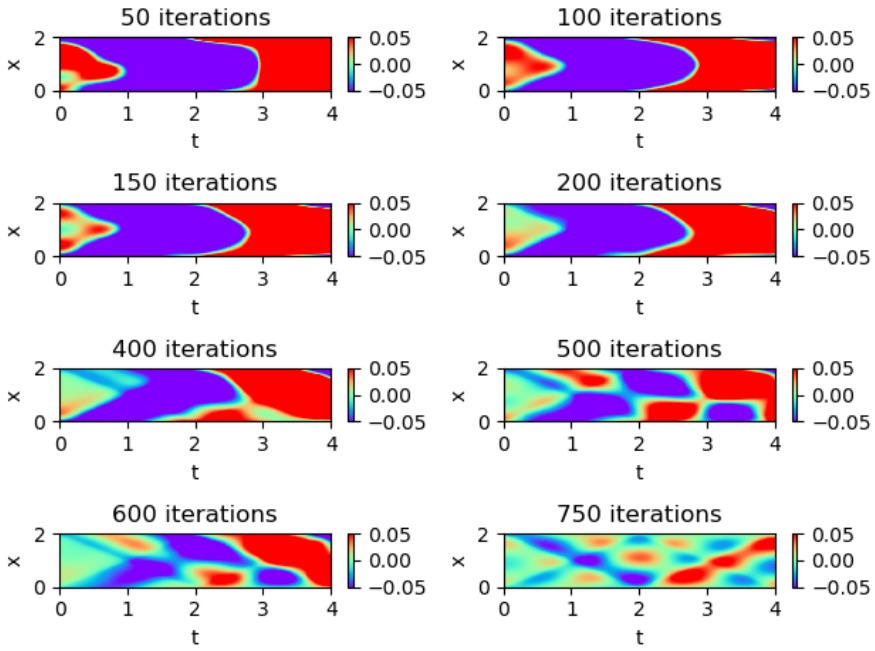


图 4.7：64-32-16-8 网络的 $\Lambda(x,\ t)$ 收敛行为。每幅图均显示相对于真实解的绝对误差。

找到合适的容差为止，损失定义为

$$\mathcal{L}^{(0)}(\theta) = \frac{(1-\lambda)}{N_d}\sum_{i=1}^{N_d}\Big(\Lambda(x_i,t_i;\theta) - u(x_i,t_i)\Big) + \frac{\lambda}{N^{(0)}}\sum_{j=1}^{N^{(0)}}\Big(\mathcal{N}(\Lambda(x_j^{(0)},t_j^{(0)};\theta))\Big) \quad (4.11)$$

其中，每个(x_i, t_i)都是已知数据点，$(x_j^{(0)}, t_j^{(0)}) \in X^{(0)}$是密集区域内的一个随机采样点，$N^{(0)}$是从$X^{(0)}$中随机采样的点数。经过预定次数的迭代 m （或达到收敛）后，定义 $X^{(1)} = X_1^{(1)} \cup X_2^{(1)} \cup ... X_n^{(1)}$，使得对于 $i \neq j$，每个 $X_i^{(1)} \supset X_i^{(0)}$ 和 $X_i^{(1)} \cap X_j^{(1)} = \varnothing$ 均满足条件，因此 $X^{(1)} \supset X^{(0)}$。定义

$$\mathcal{L}^{(1)}(\theta) = \frac{(1-\lambda)}{N_d}\sum_{i=1}^{N_d}\Big(\Lambda(x_i,t_i;\theta) - u(x_i,t_i)\Big) + \frac{\lambda}{N^{(1)}}\sum_{j=1}^{N^{(1)}}\Big(\mathcal{N}(\Lambda(x_j^{(1)},t_j^{(1)};\theta))\Big) \quad (4.12)$$

其中每个 $(x_j^{(1)}, t_j^{(1)}) \in X^{(1)}$ 是 $X^{(1)}$内的一个随机采样点，并运行至收敛或达到 m 次迭代。继续执行，直到 $X^{(k-1)} \subset X^{(k)} = X_1^{(k)} \cup X_2^{(k)} \cup ... \cup X_n^{(k)} = Y$，其中 $X_i^{(k)} \supset X_i^{(k-1)}$ 且 $X_i^{(k)} \cap X_j^{(k)} = \varnothing$ 对于 $i \neq j$。然后，使用从整个区域中抽取随机样本的常规损失函数。

$$\mathcal{L}^{(k)}(\theta) = \frac{(1-\lambda)}{N_d}\sum_{i=1}^{N_d}\Big(\Lambda(x_i,t_i;\theta) - u(x_i,t_i)\Big) + \frac{\lambda}{N^{(k)}}\sum_{j=1}^{N^{(k)}}\Big(\mathcal{N}(\Lambda(x_j^{(k)},t_j^{(k)};\theta))\Big) \quad (4.13)$$

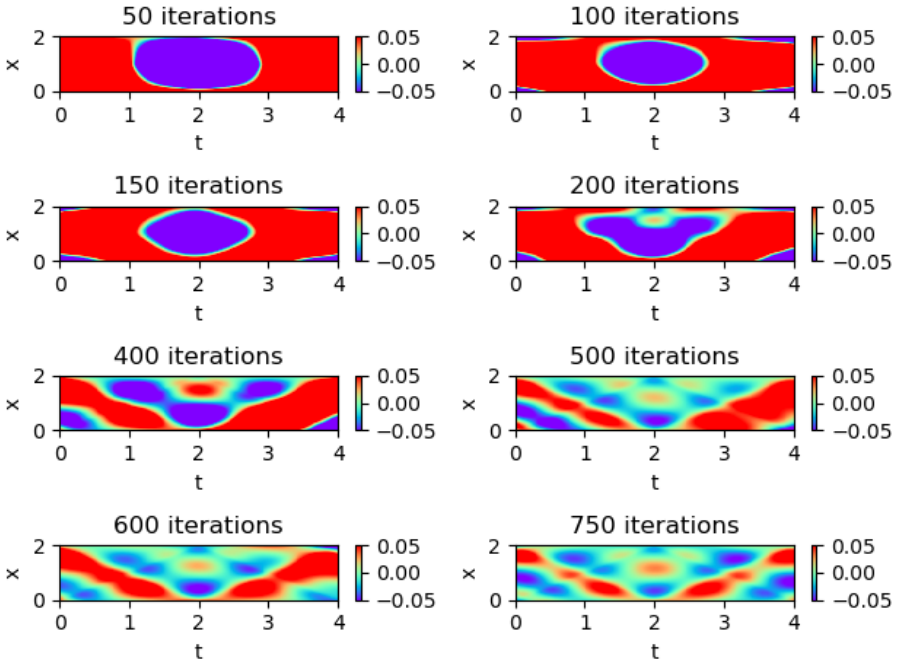


图4.8： $\Lambda(x,t)$的收敛行为，其中数据取自 $t=2$，而非直接取自初始条件。

定理 1表明，优化过程应首先专注于近似数据点附近的函数行为，然后再扩展区域，并解决数据较少位置的误差。这是因为缺乏数据的区域依赖周围行为的准确性来正确地建模解，因此，最小化密集数据区域中的误差应当是首要任务。

4.4.2 误差模式

第二点指出，绝对误差并非在整个区域内随机分布，而是呈现出一定的结构。这种结构在每个近似解中似乎都保持一致，并且不受数据位置或网络结构的影响。

由于波的物理行为相当容易凭直觉理解，因此不难看出，这些误差似乎出现在波形最为“弯曲”之处。误差似乎以交叉模式来回转换，从边界到中心，再返回边界。这可能是由导数误差造成的。

- 当 $|u(x,t)| \to 1$时，误差会出现在边界处，因为此时$\left|\frac{\partial u}{\partial x}\right|$达到最大值（波呈现出最大的空间曲率）。
- 当 $|u(x,\ t)| \to 0$ 时，波的中心会出现误差，因为此时$\left|\frac{\partial u}{\partial t}\right|$达到最大值（弦的中心正以较大的相对速度运动，

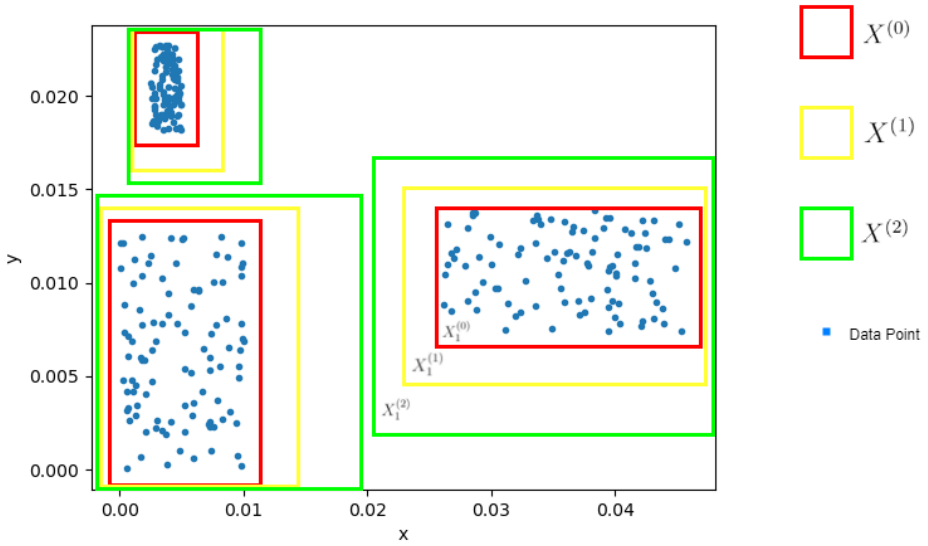


图4.9：这是二维区域数据集的一种可能数据分组方式。$X^{(i)}$ 的每次迭代都会扩展每组数据周围的可访问域。该算法会先对每个分区内部的行为进行建模，然后再扩展到区域其他部分的行为

或最大时间曲率）。

导数中的误差显然遵循近似解误差所呈现的模式。误差的相对幅度似乎也更大，并且（和之前一样）会从初始条件处向外传播；数据最密集处的整体导数估计值最佳。

这一发现颇为有趣，因为它表明，分析物理信息神经网络中发生快速变化的位置，或许能够告诉我们误差可能出现的位置，而无需知道解析解。作为检验，我们对一个拟线性偏微分方程进行了相同的分析。

拟线性偏微分方程是偏微分方程的一个特殊子类，因为它们可能形成激波[1]。当解趋近于激波点时，我们通常会发现行为发生非常剧烈的变化，甚至出现不连续性。

拟线性偏微分方程的一个著名例子是黏性伯格斯方程，该方程用于流体力学、非线性声学和气体动力学的建模。其定义为

$$\phi_t + \phi\phi_x = \nu u_x x \tag{4.14}$$

其中，ν 是黏度参数。当 ν的数值较小时，解可能会形成

[1]对于偏微分方程，当点 $u(x,\ t_s)$在某个 x 上对于其应取的数值存在冲突信息时，解中会在 t_s 处形成激波（即，对于 $t \geq t_s$而言，解变为多值）。这通常意味着，控制方程的假设在该点之后不再有效

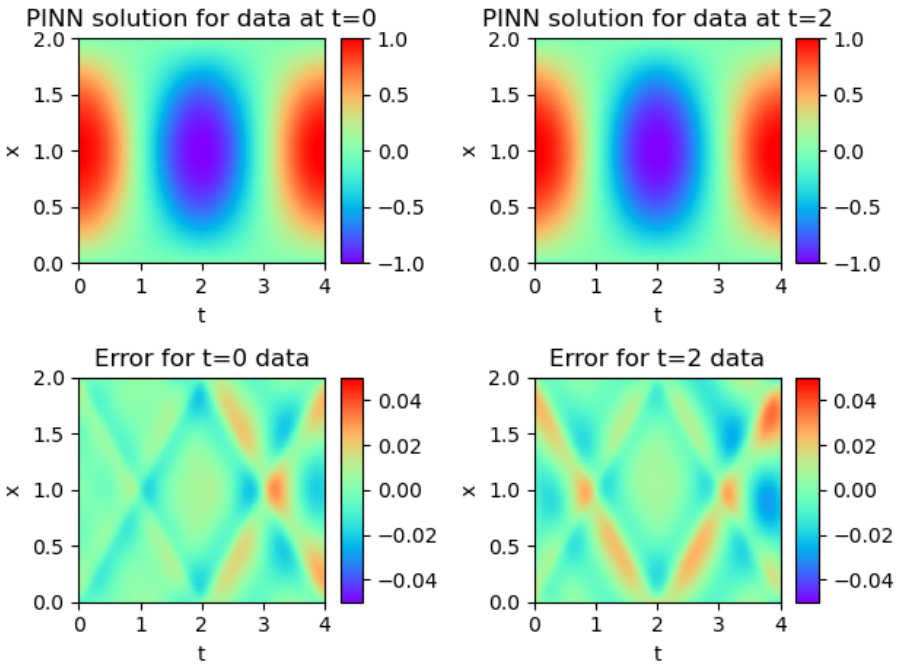


图4.10：来自不同定义域位置的数据集对应的近似解与误差模式。该模式与数据分布无关

激波，这会使系统变得极难求解。以下问题描述并求解于 Raissi, M. 等人的研究中。[49]

$$\begin{cases} \phi_t + \phi\phi_x - \frac{0.01}{\pi}\phi_{xx} = 0 & \text{for } 0 \le t \le 1, \quad -1 \le x \le 1 \\ \phi(x,0) = -\sin(\pi x) & \text{IC} \\ \phi(-1,t) = \phi(1,t) = 0 & \text{BCs} \end{cases} \tag{4.15}$$

解的数据直接取自所引用的论文。显然，近似解拟合得很好，且 L_2 误差为 4.9×10^{-4}。然而，我们关心的是误差会出现在哪里——如果确实存在误差的话。

从图 4.12 中解的形式来看，并结合一维波动方程误差位置分析中所述的逻辑，我们应该预期有2处误差区域。

1. 由于时间导数的影响， $t < 0.4$ 附近会出现 δ形状（朝右）的较小误差。此处的 t导数最大，但并不算特别大。

2. 由于空间导数较大， $t > 0.4$, $x \approx 0$ 附近会出现较大误差。事实上，该处几乎变成了间断。

Fr从图4.13可以看出，预测显然是准确的。尽管误差很小 L_2 err or

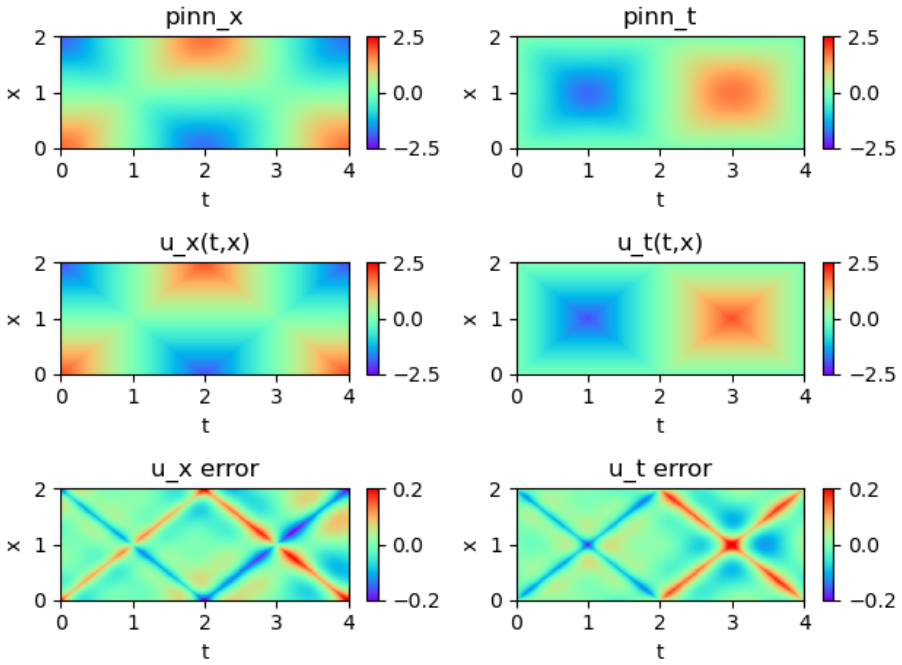


图 4.11：在 $t = 0$ 处所有数据的解析解导数与物理信息神经网络解导数之间的误差

正如研究中所记录的，误差几乎完全是由 x 方向上极其陡峭的导数造成的。这说明，通过更智能的数据采样，我们可以改进近似解。

定理 2 设 $\Lambda(x,t)$ 为一个物理信息神经网络，它能够紧密近似某个函数 $u(x,t)$ 的总体行为；设 $\epsilon > 0$ 为可接受的容差，用以判断解是否已收敛。设 m 为最大迭代次数。若在第 m 次迭代时，我们有

$$\frac{1-\lambda}{N_d}\sum_{i=1}^{N_d}||u(x_i,y_i,t_i)-\Lambda(x_i,y_i,t_i)||_2^2+\frac{\lambda}{N_s}\sum_{j=1}^{N_s}||\mathcal{N}(\Lambda(x_j,y_j,t_j))||_2^2>\epsilon \tag{4.16}$$

那么，该网络未能在有限时间内收敛到一个解。设 $\Lambda(x,t)$ 并计算

$$\mathcal{J}_\Lambda = \nabla\Lambda(x,t) \tag{4.17}$$

使得雅可比矩阵 $\mathcal{J}_\Lambda$ 是一个向量，其中包含 $\Lambda\,(x,t)$ 相对于每个变量的导数。由于当 $\mathcal{J}_\Lambda$ 相对较大时，最有可能出现误差，因此我们可以使用 $\mathcal{J}$ 作为数据采样的概率密度函数，而不是在整个区域内均匀采样。也就是说，

$$\frac{||\nabla\Lambda(x_i,t_i)||_2}{\max||\nabla\Lambda(x,t)||_2}\to 1,\quad P[(x_i,t_i)\in N_s]\to 1 \tag{4.18}$$

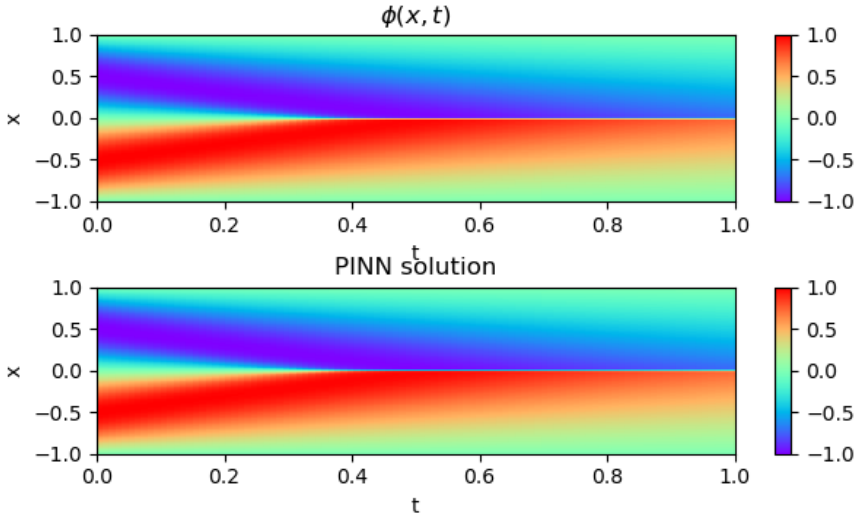


图 4.12：伯格斯方程的真实解与物理信息神经网络近似解

其中 N_s 是物理项采样数据集。

从根本上说，如果我们怀疑训练后物理信息神经网络近似可能会因导数而产生较大误差，就可以引入另一个训练步骤（或者也许可以称之为物理信息神经网络验证步骤）。由于误差很可能出现在相对变化较大的区域（导数较大），在训练后期，我们应在这些区域进行更密集的采样，然后检查物理信息神经网络在整个区域内是否达到最小值。

这里的建议是，将区域内的导数用作概率密度分布的替代。区域内某处的导数越大，我们就越有可能从该位置采样一个点。

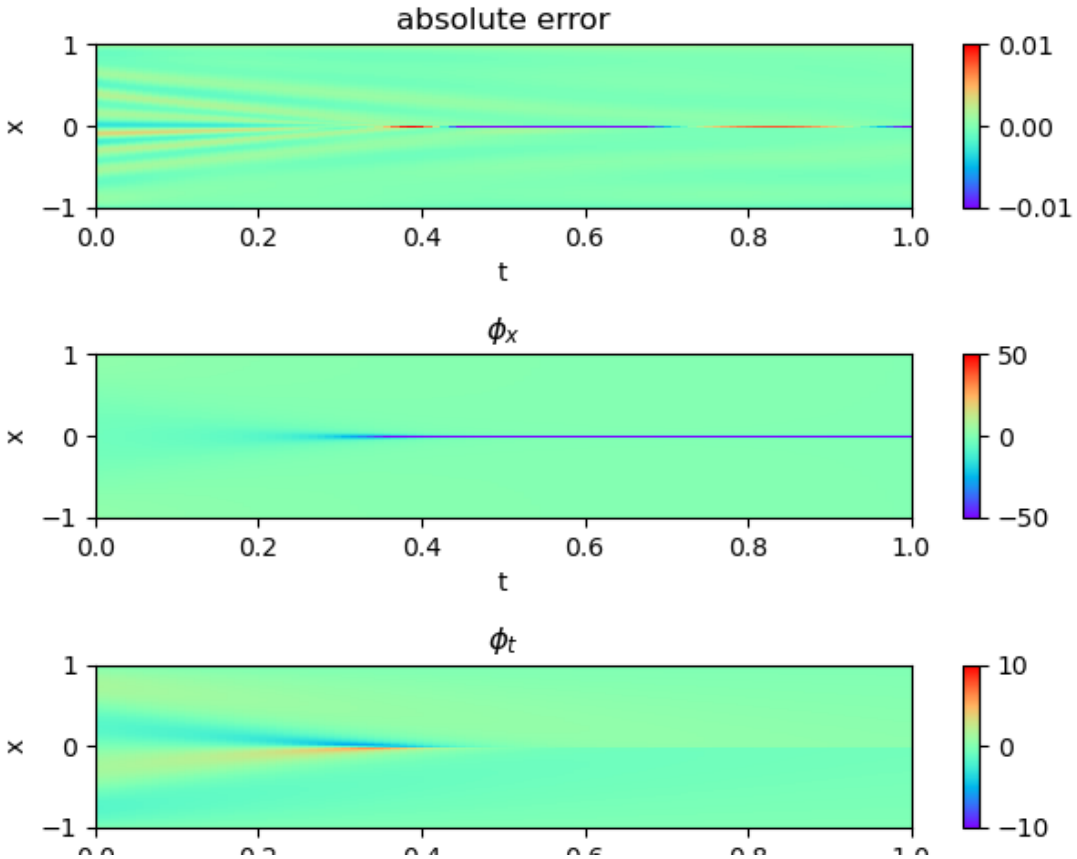


图4.13：上：解析解与物理信息神经网络近似解之间的误差。中：关于空间的导数。$x \approx 0$ 处的导数远大于其他所有导数。下：关于时间的导数。它远小于空间导数中的极值

第5章

当前应用与研究领域

物理信息神经网络已应用于学术研究，用于构建能够求解或估计物理系统的神经网络。这些问题涵盖范围广泛，既包括磁共振成像扫描 [39] 和自然语言处理[1] [40]，等小规模问题，也包括地震成像和流体动力学等大规模问题。

物理信息神经网络可用于求解偏微分方程的正演问题，即估计其解；也可用于求解反问题，即近似方程组中的函数系数。本文将探讨这两类问题。

5.1 地球物理层析成像

地球物理层析成像 是应用无损方法研究地下地形属性和结构的过程 [41]。根据区域的规模以及地质学家所需的细节程度，可以采用多种不同的技术对地下结构进行建模。

一种应用广泛且能够生成高分辨率图像的技术是地震成像，它利用在不同点测量到的强声波来构建地下参数的近似。波场重建反演（WRI）就是此类方法之一，且尤其受欢迎 [43]。

5.1.1 波场重建反演

WRI 是一种用于地震成像的替代方法，建立在正向波场反演 （FWI）中已确立的技术之上。FWI 是一种数据驱动的、受约束的非线性优化方法，利用已知的声波方程物理学

[1]自然语言处理是一种计算技术，它接收连续数据（如波长、振幅等），并对数据进行分析，使计算机能够识别并理解单词和语音

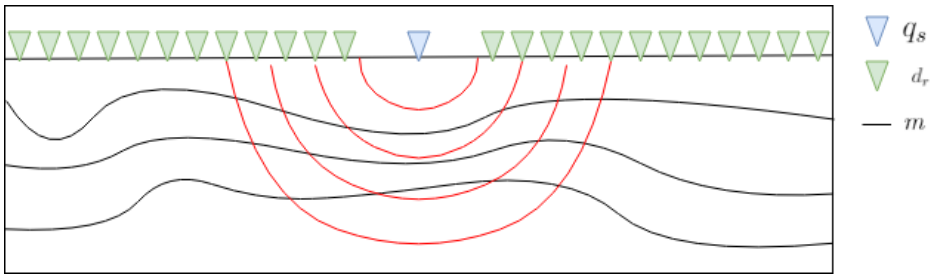


图5.1：利用地震波成像的基本示意图。波（红色）从震源 q_s发出，穿过介质 m（大多数情况下为地球），传感器d_r 测量地表上的波响应。响应会随介质结构而变化

从波动方程的部分测量数据中重建特征，具体是在频域而非时域中进行 [44]。之所以优先使用频域而不是时域，是因为不同频率的波会根据其所穿过介质的性质而表现出不同的行为。它们可以反射、折射、发生剪切并穿透。频域波（亥姆霍兹）方程为

$$(\nabla^2 + \frac{\omega^2}{c(x)^2})u(x,\omega) = q(\omega)\delta(x - x_s) \tag{5.1}$$

其中， u 表示频域中的声学地震波场， ω 表示角频率， x 表示空间位置， c 表示介质的速度， $q(\omega)\delta(x - x_s)$表示位于 $x = x_s$ 处的频域震源（因此 $\int_{-\infty}^{\infty} q(\omega)\delta(x - x_s)dx = q(\omega)$）。简言之，离散亥姆霍兹算子 A （也称为阻抗矩阵）满足

$$A(m,\omega) = L^2 + \omega^2 \text{diag}(m) \tag{5.2}$$

其中， L^2 表示离散拉普拉斯算子 ∇^2，而 m 表示介质的离散平方慢度$c \frac{1}{x}()^2$。

——

基本原理是将声波发射到地表，并由远程传感器测量波场响应，传感器通常位于地形地表上或附近。每个传感器 d_r 都会采集来自每个声学震源 q_s的数据点。

利用这些数据，可以得到一个约束优化问题，使得

$$\min_{m,u} \frac{1}{2} \sum_{\omega} \sum_{s,r} ||\Gamma_{s,r} u_s(\omega) - d_{s,r}||_2^2 W_{s,r} \qquad \text{s.t. } A(m,\omega)u_s(\omega) = q_s(\omega) \tag{5.3}$$

其中，$W_{s,\ r}$ 是施加于数据残差的权重算子。可通过使用内积和拉格朗日乘子来消除该约束条件，这称为伴随状态

或降维空间表述，使得

$$\mathcal{L}(m, u_s, v_s) = \frac{1}{2}\sum_{s,r} ||\Gamma_{s,r}u_s - d_{s,r}||_2^2 W_{s,r} + \sum_s \langle v_s, A(m)u_s - q_s\rangle_x \tag{5.4}$$

其中 $\langle\cdot,\cdot\rangle_x$ 是空间坐标上的内积， v_s 是拉格朗日乘子。对每个最小化参数求偏导，得到

$$\begin{aligned}
\frac{\partial\mathcal{L}(m, u_s, v_s)}{\partial m} &= \sum_s \left(\frac{\partial A(m)u_s}{\partial m}\right)^* v_s \\
\frac{\mathcal{L}(m, u_s, v_s)}{\partial u_s} &= \sum_{s,r} \Gamma^*_{s,r}W^*_{s,r}(\Gamma_{s,r}u_s - d_{s,r}) + \sum_s A^*(m)v_s \\
\frac{\partial\mathcal{L}(m, u_s, v_s)}{\partial v_s} &= \sum_s A(m)u_s - q_s
\end{aligned} \tag{5.5}$$

其中 $*$ 是共轭转置。最后一个方程意味着，如果我们处于最小值处，那么 $A(m)$ $u_s = q_s$，因此原最小化问题中的约束条件得以满足。简言之，每次迭代求解波动方程和伴随方程即可得到梯度，继而可用其计算局部极小值。

然而，这类反演存在三个主要问题：

1. 每次迭代都要对整个空间求解两次波动方程，计算开销很大，而且构造出的海森矩阵通常是稠密的。

2. 对地球模型 m存在极强的非线性依赖。

3. 该问题是非凸的，因此解极少是唯一的。因此，找到的最优解取决于地球模型 m 的初始参数。换言之，局部极小值确实会造成问题。这通常称为周跳，当模型 $\Gamma_{s,\ r}u_s$生成的数据与记录的数据 $d_{s,\ r}$相差超过半个周期时，就会发生这种情况。

WRI 通过赋予目标函数更多自由度来解决这个问题。它通过权重 λ^2 [43] 略微放宽物理约束条件，并将其作为惩罚参数直接加入目标函数。

$$\min_{m,u_s} \mathcal{L}_\lambda(m, u_s) = \sum_{s,r} \left(||\Gamma_{s,r}u_s - d_{s,r}||_2^2 + \lambda^2||A(m)u_s - q_s||_2^2\right) \tag{5.6}$$

换句话说，我们认为，最优地球模型解应在物理学与数据之间取得平衡，而非完全符合物理学。这消除了求解伴随波动方程的需要；如果 λ 较小，海森矩阵便是稀疏的（ λ 较小则意味着对物理学的依赖程度较低） [44]。近来，波动方程反演已被应用于时域问题，而非频域，以

降低一些计算成本（因为可以更轻松地利用 LU 分解求解多个线性系统） [45]。

5.1.2 使用物理信息神经网络求解波动方程

波场反演的两大基石是

1. 构建满足源数据和接收器数据的波场。2. 利用估计出的波场重建地下边界下方的地形

正如牛津大学在利用物理信息深度学习求解波动方程 （B. Mosely 等人）[46]中所展示的，物理信息机器学习在准确重建复杂区域中的波场方面展现出巨大潜力。该论文应用第 2 章和第 3 章中介绍的方法和策略，估计波在复杂程度各异的介质中传播时产生的声学响应。

方法

该研究使用二维声学波动方程（如方程 3.1 所述）构建损失函数

$$\mathcal{L} = \frac{1}{N_u}\sum_{i=1}^{N_u} ||u(t_i, x_i, s_i) - \Lambda(t_i, x_i, s_i)||^2 + \frac{1}{N_\Lambda}\sum_{j=1}^{n_\Lambda} ||\mathcal{N}(\Lambda(t_j, x_j, s_j; \theta); \lambda)||^2 \tag{5.7}$$

其中，$(t_i,\ x_i)$ 是已知的初始值，$(t_j,\ x_j)$ 是在整个区域内采样的点，Λ 是物理信息神经网络，λ 是物理项最小化的权重函数，θ 是定义 Λ 的可训练参数，而 $\mathcal{N}$ 是描述声波方程的微分算子，具体定义见方程 3.3。令人惊讶的是，作者并未使用自动微分来准确计算损失函数中物理项所需的梯度，不过作者确实指出，这种方法可以（而且确实应该）使用。

本研究并未探讨网络架构，以及不同架构可能如何改变解。受 Raissi 等人（2019）的启发，所使用的激活函数 $\sigma:\ \mathbb{R} \to [0, \infty]$ 为 SoftPlus 激活函数

$$\sigma(x) = \ln(1 + e^x) \tag{5.8}$$

且其导数等于逻辑映射 $\sigma'(x) = \frac{1}{1+e^{-x}}$。使用它通常是因为它能降低梯度消失的可能性。PINN 的结构固定为 10 层，每层包含 1024 个隐藏通道，并采用全连接结构。输入空间由空间坐标 $x \in \mathbb{R}^2$、时间坐标 $t \in \mathbb{R}$ 和源项 s组成，

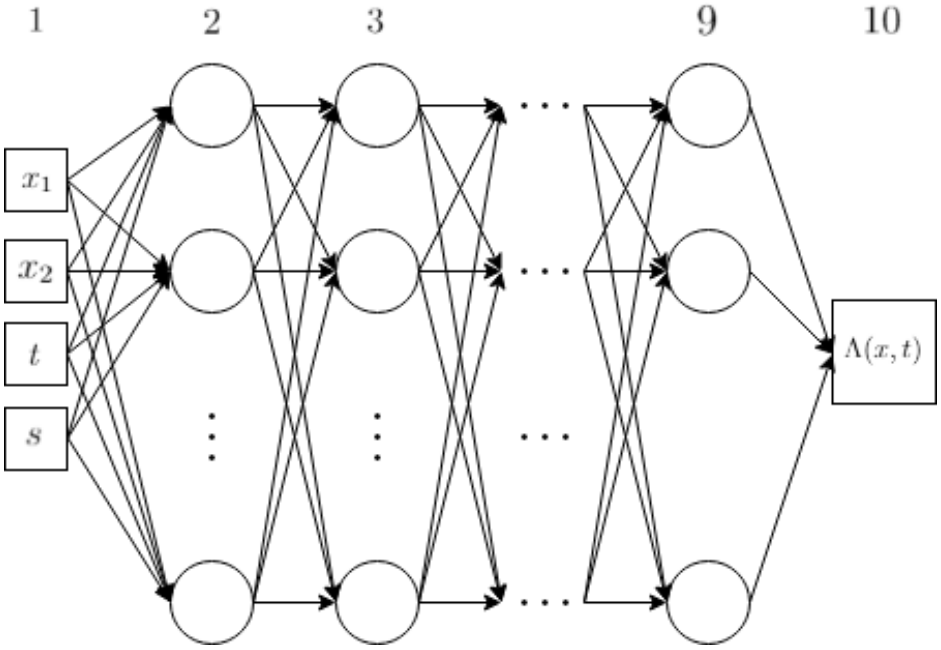


图 5.2：求解波动方程的网络架构。该网络由 10 层组成，共有 1024 个节点。最终输出层采用线性形式，而不是使用非线性激活函数。

输出为波场 $u \in \mathbb{R}$。除最后一层外，每一层都应用了激活函数；最后一层则保持线性（否则网络输出将始终为 $\Lambda(x,\ t) > 0$）。

该研究使用 ADAM 优化算法计算网络权重和偏置的最小值。

Data

该研究需要两类数据：初始波数据和介质速度数据。由于该研究仅关注波场建模的正演问题，因此介质密度和速度事先已知。该研究使用了 3 个复杂程度各不相同的数据集。

数据集 1 是恒定速度介质；数据集 2 的介质速度在垂直方向上空间变化（即伪层状速度）；数据集 3 来自著名的 Marmousi 模型[2], ，其速度在所有空间方向上均发生变化。

为了计算足够的初始条件用于训练，前 T_1 个训练步使用有限差分法来生成离散波场。因此，边界

[2]Marmousi 模型数据集创建于 1988 年，现已成为测试地震成像技术的行业标准数据集。该数据由法国石油研究院（Institut Francais du P´etrole）最初创建，是合成数据，基于宽扎盆地北昆圭拉槽的地下地形结构。

数据来自集合 $x_i \in [0, X_{\max}]$ 和 $t_i \in [0, T_1]$，其中 $T_1 \ll T_{\max}$。对于常量数据集和分层数据集，$T_1 = 0.02$ 秒，对应最初的 10 个时间步。对于 Marmousi 数据集，$T_1 = 0.04$ 秒，因此使用了最初的 20 个时间步，以及来自多个震源的多次模拟。

训练分为两个不同阶段：边界阶段和物理阶段。第一阶段使网络能够优先最小化边界损失项，而不受物理损失的干扰。迭代进行到一半时（第 500'000 步），引入物理损失项，使网络能够在区域的其余部分进行最小化。

结果

针对每个数据集都进行了完整的有限差分模拟，作为"真实值"；误差则是有限差分模型与物理信息神经网络模型之间的差异。本研究旨在考察物理信息神经网络对已知二维速度区域中的波传播进行建模的效果，以及其表现比标准人工神经网络好多少。

标准人工神经网络通常能够捕捉向外传播的波，但往往无法捕捉剪切波、反射和折射等更微妙的现象，因此无法将解泛化到训练区域之外很远的地方（即 $t > T_{\max}$）。引入物理损失后，这一问题显著减轻。

人工神经网络方法中的误差往往出现在断层线处（即速度发生显著变化的位置）。这可能是因为没有向人工神经网络提供关于介质速度变化时波场应如何随时间和空间变化的信息，因此在缺乏充足数据的情况下，它无法学习反射波等的特性（而这些数据在实际应用中往往缺失）。

作者强调，使用物理信息神经网络相比有限差分模拟的最大优势在于，其解具有更强的泛化能力。例如，在第三个数据集中，使用多个震源创建了一个物理信息神经网络。如果想要估计同一区域内新震源在某个时刻（例如 $t = 0.4$）产生的波场，有限差分方法就需要使用

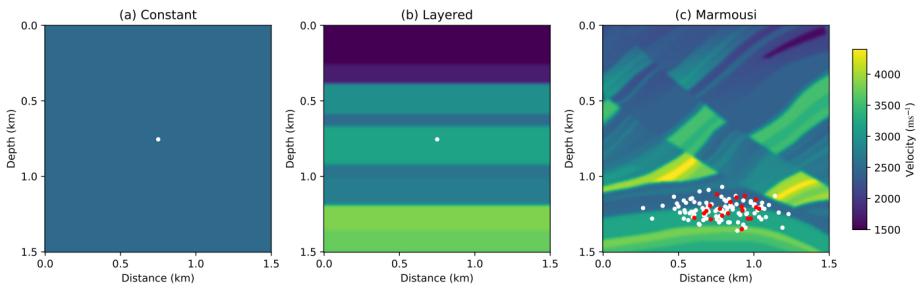


图 5.3：描述介质速度的不同数据集。白色点为产生声波的震源

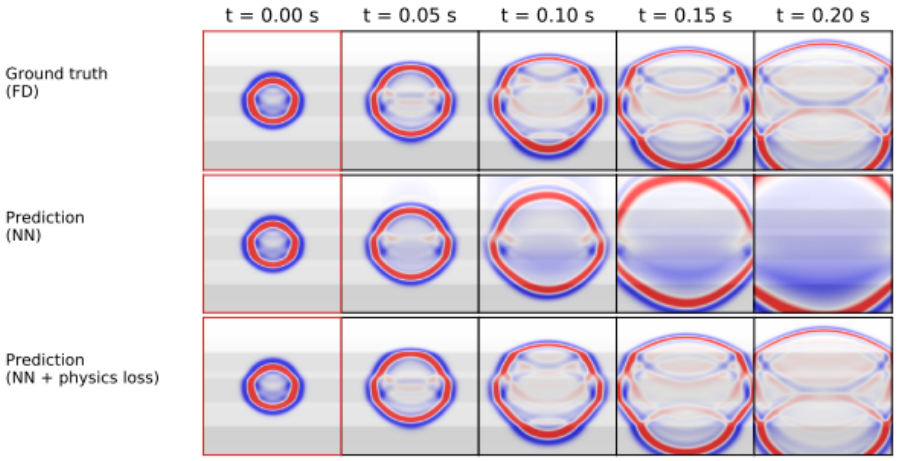


图 5.4：使用三种方法（有限差分方法、人工神经网络、物理信息神经网络）对分层数据集中的波动行为进行估计。人工神经网络未能捕捉较大t处的许多反射波行为，但确实模拟了前导波动行为

新的震源项，模拟所需时刻之前所有时间、整个空间中的情况。由于物理信息神经网络解的泛化能力更强，无需进行额外模拟即可得到该数值，只需输入所需的时间、空间和源信息即可。这意味着，使用训练后的物理信息神经网络比使用有限差分方法快约 $\times 1000$ 倍。

物理信息神经网络方法也很容易扩展，因为增加维度、更换数据集以及改变物理学都非常简单。然而，介质速度中的不连续性仍可能导致解出现误差，尽管有限差分方法也存在这一问题。为避免这个问题，在第三个数据集中，对两种不同速度的交界处进行了轻微平滑。这个问题有三种可能的解决方案：

1. 为地下地形的每个区域分别使用物理信息神经网络，从而允许介质中存在不连续性。

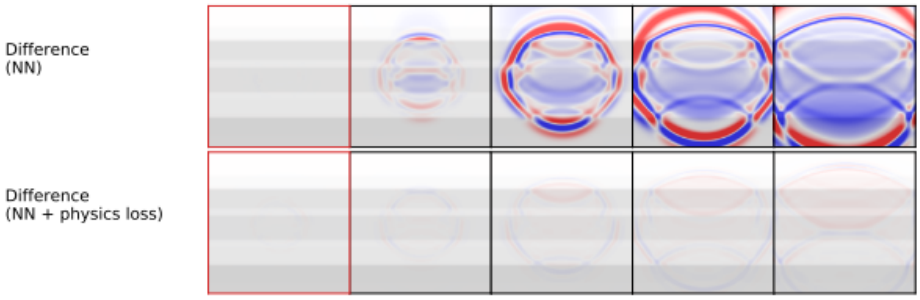


图 5.5：有限差分模拟与两个网络各自之间的差异。物理信息神经网络与有限差分模拟之间的误差几乎可以忽略不计

2. 引入一个允许波场分离的损失项。

3. 对数据采用更智能的采样方案，使速度变化较大的区域在最小化过程中被更频繁地采样，正如上一章所讨论的那样。

不过，结论是，物理信息神经网络能够学习波所表现出的许多复杂行为，能够生成连续解，在训练数据范围之外也能准确估计数值，而且训练完成后使用起来非常快捷。

5.1.3 反问题

既然已经证明物理信息神经网络能够为波场求得准确的解，那么更关键的问题是，它们能否用来替代传统方法求解反问题——确定介质速度的分布。阿卜杜拉国王大学的 Chao Song 等人（2021） [47] 将第 3 章所述的物理信息神经网络应用于改进WRI方法。

方法

这项研究的核心思路是使用两个物理信息神经网络来重建速度场。

1. T第一个 PINN 按频率重建了散射波场 s.

2. 第二个 PINN 使用第一个 PINN 重建的波场来修改速度场（平方慢度）。

研究没有使用时域波动方程，而是采用了声波方程的 Lippmann–Schwinger 形式 [48]

$$\omega^2 m\delta u + \nabla^2 \delta u = i\omega^2 \delta m u_0 \tag{5.9}$$

该形式可求得不同角频率 ω下的散射波场 $\delta u = u - u_0$，其中 u 为波场，且

$$u_0(x) = iH_0^{(1)}(\omega\sqrt{m_{n0}}|x - x_s|) \tag{5.10}$$

对于各向同性二维区域， 是背景波场，其中 x_s 是点源位置。 $\delta m = m - m_0$ 是扰动平方慢度，其中 m_0 是均匀地球模型的慢度。 $H_n^{(k)}$ 是第二类汉克尔函数，定义为

$$H_0^{(1)}(z) = \frac{1}{i\pi}\int_0^\infty \frac{e^{(\frac{z}{2})(\frac{t-1}{t})}}{t}dt \tag{5.11}$$

它包含实部和虚部。

使用这组方程，正演问题的物理信息损失函数（在这篇文献中称为均方误差）按物理约束与数据之间的平衡这一标准方式定义

$$\begin{aligned}\mathrm{MSE}_f = \sum_{is}^{N_s}\left(\frac{1}{N_r}\sum_{i=1}^{N_r}||\delta d^i - \delta u(x_r^i)||_2^2 + \right.\\ \left.\frac{\alpha}{N}\sum_{i=1}^{N}||\omega^2 m_1^i \delta u^i + \nabla^2\delta u^i + \omega^2(m_1^i - m_0^i)u_0^i||_2^2\right)\end{aligned} \tag{5.12}$$

其中，第一项是数据损失项，第二项则通过利普曼–施温格方程概括了系统的物理特性。方程中的可调参数决定了散射波场 δu 的行为。每个 is 都是一个震源位置，$\delta d = d^i - u_0(x = x_r)$ 是记录到的散射数据，m_1 是初始平方慢度模型。 α 是权重项，通过反复试验将其设为 0.00001； N 是为物理约束选取的点数。

反问题的物理信息损失函数定义如下

$$\begin{aligned}MSE_b = \frac{1}{N_p}\sum_{i=1}^{N_p}||\omega^2 m^i \delta u^i + \nabla^2\delta u^i + \omega^2(m^i - m_0^i)u_0^i||_2^2 + \\ \epsilon\sqrt{\left(\frac{\partial m^i}{\partial x}\right)^2 + \left(\frac{\partial m^i}{\partial z}\right)^2}\end{aligned} \tag{5.13}$$

值得注意的是，该函数没有数据项。第二项称为全变差项（TV），加入该项是为了稳定训练过程，其中 $\epsilon = 0.1$。这里，散射波 δu 是固定频率下的物理信息神经网络解。反演物理信息神经网络利用正演物理信息神经网络，进而对地球模型 m 进行调整。实际上，该算法看似简单。

因此，通过对不同频率的散射波场进行建模，速度模型得以持续改进。

由于任务复杂、区域范围较大且区域具有动态特性，本研究使用的网络规模相当大。正向（散射波场）物理信息神经网络采用递减层级结构，包含 8 个隐藏层，排列顺序为 $[128, 128, 64, 64, 32, 32, 16, 16, 8, 8]$,；而逆向（速度）物理信息神经网络采用扁平结构，包含 8 个隐藏层，排列顺序为 $[20, 20, 20, 20, 20, 20, 20, 20]$。散射波场物理信息神经网络的输入为 $\mathbb{R}^3$，包含一个二维位置 x 和一个源项x_s；其输出为 $\mathbb{R}^2$ ，即散射波场的实部和虚部。速度模型物理信息神经网络的输入是一个二维点，输出则仅为速度。

网络使用 ADAM 优化算法进行了 20’000 次迭代训练

算法3 基于PINN的WRI

要求： d ▷ 观测数据集
要求： m_0 ▷ 背景平方慢度模型
要求： m_1 ▷ 初始平方慢度模型
要求： u_0 ▷ 背景波场
要求： n ▷最大迭代次数
要求： $[f_{\min}: \ df: \ f_{\max}]$ ▷ 选定的角频率
要求： α ▷ 正演求解器中物理项的权重
对于 $f = [f_{\min}: \ df: \ f_{\max}]$ **执行**
对于 $i = 1: \ n$ **执行**
重建固定频率下的散射波场
使用重构的散射波场重建速度模型
结束循环
结束循环
返回 反演速度模型

算法，随后进行了 150’000 次 L-BFGS 迭代。

Data

该方法在两个数据集上进行了测试，但我们将重点关注二维 Marmousi 模型，原因有二：

1. 该数据集曾用于先前研究，因此便于进行比较。2. 它是两个数据集中更复杂的一个，因此更值得分析。数据包括：1. 10 个震源，沿地表等间距分布。2. 301 个接收器，位于深度 25 m 处等间距分布，用于收集数据损失项的数据。3. 区域内随机采样的 80’000 个点，用于物理损失。

初始速度模型的生成方式使得速度随 z 的增加而增大（因此是一个分层模型，与先前研究中使用的第二个数据集并无太大差异）。

结果

针对 3 赫兹、4 赫兹和 5 赫兹（3 组角频率）执行基于PINN的WRI后得到的速度场，生成了一个具有真实解某些特征的地球模型，但模型过于平滑。这是由实验数据造成的

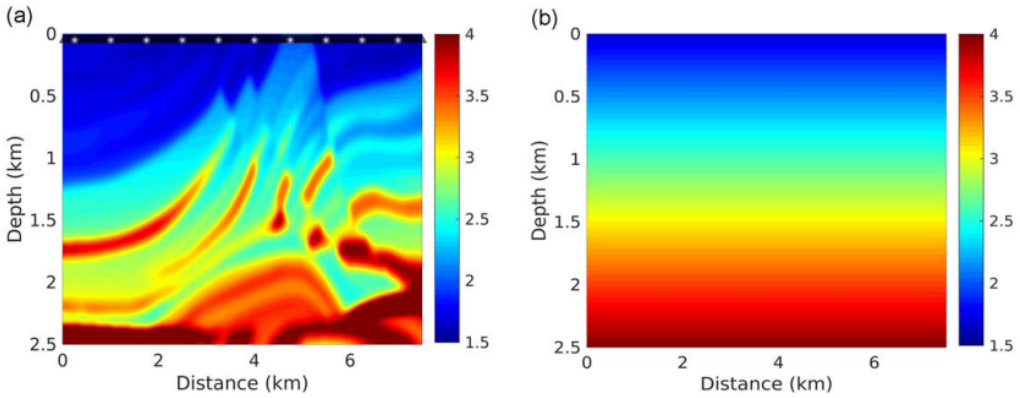


图 5.6：左：真实的 Marmousi速度模型。右：PINN WRI算法的初始速度模型

这些数据已在上一章中进行分析。这里的误差是由于散射波场过于平滑所致。换言之，导数变化较大的区域往往误差也较大。然而，尽管基于PINN的WRI得到的速度模型并不能很好地近似真实解，但它确实证明了自身极其有用。

第 5.1.1 节详细讨论了全波形反演和波动方程反演的问题与不足。其中一个不足是，最小化问题往往受到局部极小值的困扰，最终解取决于所提供的初始地球模型。构建一个良好的初始地球模型极具挑战，往往甚至无法实现。因此，通常会使用分层初始模型（如图 5.6 所示）。

基于PINN的WRI所得到的解虽然过于平滑，无法呈现高分辨率细节，但确实捕捉到了真实二维 Marmousi 模型的一些总体行为。它显然比初始速度模型拟合得更好。随后，将基于PINN的WRI生成的最终速度模型用作全波形反演的初始地球模型，并将其解与使用分层数据作为初始模型时得到的解进行了比较。

使用 PINN 波动方程反演的解作为初始模型，比使用分层模型得到了明显更好的最终速度模型。结果看起来仍然有些过于平滑，但解的主要特征确实与真实数据相符。

在将 PINN 方法与其他数值技术进行比较时，有观点指出，尽管有限差分方法在小规模问题上运行很快，但并不适用于大规模、高维问题。这是因为有限差分方法无法像 PINN算法那样进行并行计算。此外，增加区域属性的复杂性（例如采用垂直横向各向同性介质或各向同性弹性介质）会大幅增大有限差分方法所使用的阻抗矩阵。

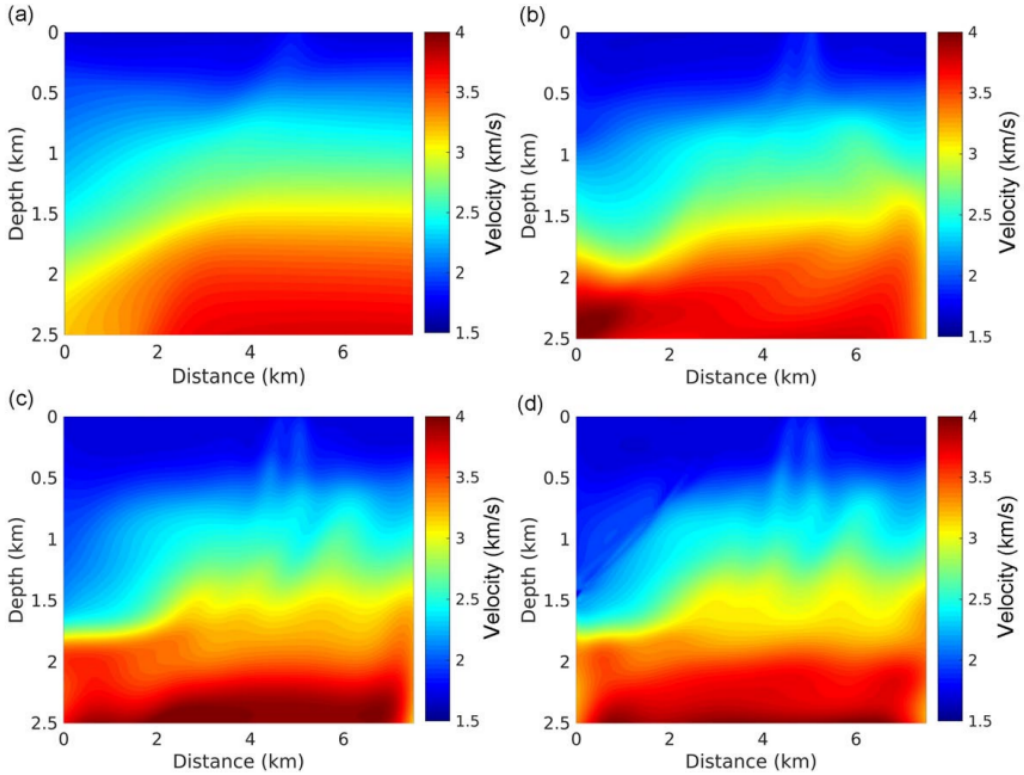


图 5.7：左上：使用 3 赫兹进行 1 次迭代后的 PINN 速度模型。右上：使用 3 赫兹进行 4 次迭代后的 PINN 速度模型。左下：使用 4 赫兹进行 4 次迭代后的 PINN 速度模型。右下：使用 5 赫兹进行 3 次迭代后的 PINN 速度模型

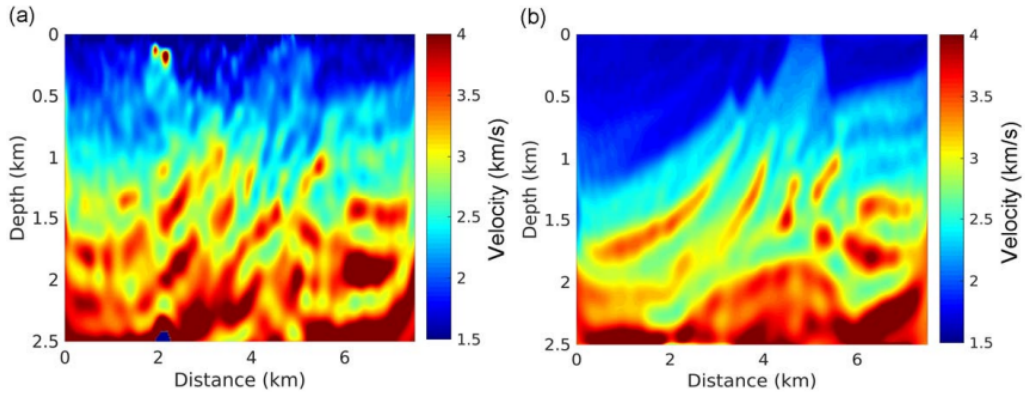


图 5.8：左：使用全波形反演从分层初始速度模型得到的估计速度模型。右：使用全波形反演从 PINN 解初始速度模型得到的估计速度模型。

最后，正如先前研究中提到的那样，这种方法的灵活性不容低估。要使波动方程反演和有限差分方法适用于更高维的区域（其中地下结构更加复杂，例如存在流体饱和的多孔区域），就需要对算法进行大幅修改；而物理信息神经网络方法只需调整损失函数，使其纳入不同的方程即可。

这里的主要缺点是，散射波场解过于平滑，进而导致反演速度模型也过于平滑。一种解决方案是适当增大网络规模，从而增加自由度；或者为每个频率分别使用一个网络（这样还可以并行处理）。不过，这显然也会大幅增加计算成本。

第 6 章

结论

由于 Python 中 TensorFlow 和 PyTorch 等软件包，以及 Julia 中 Flux 的出现，创建人工神经网络变得前所未有地简单。此外，内存的成本已大幅降低；而且，由于使用神经网络时许多运算具有并行特性，因此可以轻松利用大型高性能 GPU，大幅缩短计算时间。因此，与几年前相比，如今为具有大量参数的高维问题最小化损失函数，速度显著更快。

得益于计算微分领域的进步，研究表明，人工神经网络的损失函数可以通过特定方式构建，从而在区域中没有数据的部分，仍能准确地近似物理系统的解。此外，这些通用逼近器 还可以经过巧妙设计，以便显式利用已知行为，例如初始条件和边界条件。

物理信息神经网络生成的解是连续函数。因此，与其他数值方法不同，它们具有一项突出优势：能够为区域之外或已知数据范围之外的输入生成数值，而且无需进行任何额外计算来细化解（例如，在有限差分法中采用更小的步长，或在有限元分析中使用更精细的网格）。

此外，对误差分析的深入研究揭示了物理信息神经网络逼近中可能出现误差的方式、原因和位置。这些误差通常由以下两个原因造成：

1. 最近的一簇密集数据距离相对较远。
2. 该区域这一部分的相对导数较大。

这些误差既出现在第4章的理想波动系统中，也出现在第5章的波场重建反演法中。因此，我们可以估计近似解中误差较大的点，并有望利用数据采样技术，通过物理损失项提高这些区域的精度。这种高导数和高误差的模式

在近期研究中是一致存在的，无论是在正演问题还是反问题中都是如此，也是使用物理信息神经网络时的主要局限

并不建议用物理信息神经网络取代求解偏微分方程的传统方法，而是将其作为另一种可用于解决问题、甚至验证解的工具。正如第5章所示，物理信息神经网络解在解决现实问题时确实有其用武之地；要确定其应用范围，还需要进行更深入、更广泛的实验。这些网络在复杂系统中尤其有用，因为它们极为灵活，也易于修改，能够纳入新的物理学、新的数据或新的维度。

误差分析和收敛速度方面显然还需要进一步研究。本论文为此奠定了基础，但仍需要更严谨的证明和理论。更深入地理解解是如何收敛得到的，或许能够避免局部极小值等问题；这类问题已困扰着我们如今面临的许多优化问题。误差显然会以预先确定的方式表现出来，因此在最小化过程中就可以更有效地应对，而不是事后处理。

参考文献

[1]Samuel, Arthur L. (1959). ‘使用跳棋游戏进行机器学习的一些研究’. IBM《研究与开发杂志》. 44: 206–226. doi:10.1147/rd.441.0206[2] Rosenblatt, F. (1958). ‘感知器：一种用于大脑信息存储的概率模型’. 《心理学评论》, Vol. 65, No. 6: 386–408.[3] Swetz, Frank J. (2017). ‘数学珍宝：莱布尼茨关于微积分的论文——积分学, 收敛, 美国数学协会: 292-300.[4] Archibald, T. Fraser, C. Grattan-Guiness, I. (2004) ‘微分方程史，*1670–1950*’. 欧洲数学学会, Vol. 1, Issue 4: 2729- 2794. doi: 10.4171/OWR/2004/51[5] Babuska, I. Szabo, B. (1991) ‘有限元法导论’. Wiley: 1-50.[6]贝叶斯, T. (1763) ‘一篇旨在解决机会学说中某个问题的论文’. 英国皇家学会出版社, Vol. 53, doi: https://doi.org/10.1098/rstl.1763.0053[7]Revuz, D. (1984) ‘马尔可夫链’. 爱思唯尔, Vol. 11: 8-39, doi: https://doi.org/10.1016/S0924-6509(08)70188-0[8] Cybenko, G. (1989) ‘通过*S*形函数的叠加进行近似’. Springer, 控制、信号与系统数学 Vol. 2: 303-314, doi: https://doi.org/10.1007/BF02551274[9] 图灵, A. M. (1950) ‘计算机器与智能’. Springer, 《心智》 Vol. 59, Issue 236: 433-460, doi: https://doi.org/10.1093/mind/LIX.236.433[10]L. Rabiner and B. Juang. (1986) ‘隐马尔可夫模型简介’. EEE ASSP Magazine, vol. 3, no. 1: 4-16. 433-460, doi: 10.1109/MASSP.1986.1165342

[11] Boutros1, F. Damer, N. Kirchbuchner, F. Kuijper, A. (2021) '*ElasticFace*：用于深度人脸识别的弹性边距损失'. doi: arXiv:2109.09416[12]Grigorescu, Sorin, Trasnea, Bogdan, Cocias, Tiberiu, Macesanu, Gige (2020) '自动驾驶深度学习技术综述'. Wiley, 《野外机器人学报》，第 37 期：362–386 doi: arXiv:http://dx.doi.org/10.1002/rob.21918[13]Perlich. Claudia. Dalessandro. Stitelman, B. Raeder, O. Provost, T. F. (2013) '定向展示广告的机器学习：迁移学习实践'. Springer, 机器学习（在线版），第 95 卷：362–386 doi: 10.1007/s10994-013-5375-2[14] Baum, E. B. (1988) '关于多层感知机的能力'. 爱思唯尔，《复杂性杂志》，第 4 卷：193–214 doi: 10.1016/0885-064X(88)90020-9[15] Zhu, W. Miao, J. Qing, L. (2013) '极限学习机 *2013*：算法与应用'. Springer, 极限支持向量回归：25-28[16] Rauch, Jeffrey (1997) '偏微分方程'. Springer, 数学研究生教材，第 128 卷：1–2[17] Mamun, Abdulla - Al (2018) '抛物型偏微分方程一维热方程解析解的研究及其在计算机编程中的实现'. 《国际科学与工程研究杂志》，第 9 卷，第 9 期：913-921[18] Sobolev, S. L. (1964) '第 *1* 讲——基本方程的推导'. 数学物理偏微分方程：1-21 doi: https://doi.org/10.1016/B978-0-08-010424-9.50007-X[19] Kirkwood, J, R. (2013) '傅里叶级数'. 数学物理中的偏微分方程：Elsevier：187-212, doi: https://doi.org/10.1016/B978-0-12- 386911-1.00004-5[20] Miller, P. D. Perry, P. A. (2019) '非线性色散偏微分方程与逆散射'. 数学物理中的偏微分方程：Elsevier：doi: 10.1007/978-1-4939-9806-7[21]Fefferman, C. L. '纳维–斯托克斯方程的存在性与光滑性'，问题描述可通过克雷数学研究所获取[22] Stynes, M. O' Riordan, E. Gracia, J. L. (2018) '分级网格上时间分数阶扩散方程有限差分方法的误差分析'. 工业与应用数学学会，第 55 卷，第 2 期：1057-1079

[23] Utku, S. Melosh, R. J. (1984) '有限元分析中的解误差'. 爱思唯尔, 《计算机与结构》, 第 18 卷, 第 3 期: 379-393 1057-1079, doi: https://doi.org/10.1016/0045-7949(84)90058-0

[24]Wunderlich, W. Cramer, H. Steinl, G. (1998) '考虑局部化现象的相关与非相关塑性自适应有限元方法'。爱思唯尔, 《应用力学研究》, 第47卷: 293-308, doi: https://doi.org/10.1016/S0922-5382(98)80016-0

[25] Berry, M. W. Mohamed, A. Bee, W. Y. (2020) '面向数据科学的监督学习与无监督学习'。Springer: v-vi

[26] Janocha, K. Czarnecki, W. M. (2017) '关于分类中深度神经网络的损失函数'。爱思唯尔, DeepMind: 1-10, doi: arXiv:1702.05659

[27]Shewchuk, J. R. (1994) '轻松理解共轭梯度法'。卡内基梅隆大学, 计算机科学学院, 第1版

[28] Sakata, S. Ashida, F. Zako, M. (2010) '使用克里金法和神经网络逼近对梯度与 *Hessian* 估计进行比较研究'。爱思唯尔, 《数学与计算机建模》, 第 51 卷: 第 3-4 期: 309-319, doi: https://doi.org/10.1016/S0922-5382(98)80016-0

[29]Milan, M. Militky, J. (2011) '非线性回归模型'。Woodhead Publishing India, 《统计数据分析》: 667-762, doi: https://doi.org/10.1533/9780857097200.667

[30] Ruder, S.（2017）《梯度下降优化算法概述》。数据分析洞察中心, doi: arXiv:1609.04747[31] Sakata, S. Ashida, F. Zako, M.（2007）《一种随机逼近法》。数学统计年刊, 第22卷: 400-407, doi: 10.1214/aoms/1177729586[32] Kingma, D. P. Ba, J.（2017）《*Adam*: 一种随机优化方法》。doi: arXiv:1412.6980[33]Qina, N.（1998）《梯度下降学习算法中的动量项》。爱思唯尔, 《神经网络》, 第12卷, 第1期: 145-151, doi: https://doi.org/10.1016/S0893-6080(98)00116-6[34]Xiao, Y. Wei, Z. Wange, Z.（2008）《用于大规模无约束优化的有限内存*BFGS*型方法》。爱思唯尔, 《计算机与应用数学》, 第56卷, 第4期: 1001-1009, doi: https://doi.org/10.1016/j.camwa.2008.01.028

[35] Sherman, J, Morrison, W. J.（1950）《对应于给定矩阵中一个元素变化的逆矩阵调整》。《数学统计年刊》，第21卷，第1期：124-127，doi：10.1214/aoms/1177729893[36] Shi, Z. J. She, J.（2003）《一种采用非精确线搜索的梯度相关算法》。爱思唯尔，《计算与应用数学杂志》，第170卷：349–370，ISSN：0377-0427[37] Asl, A. Overton, M. L.（2017）《采用*Armijo-Wolfe*线搜索的梯度法在一类非光滑凸函数上的分析》。爱思唯尔，《优化与控制》：5-6，doi：arXiv:1711.08517v2[38] Liu F., Yang M.（2005）《人工神经网络模型的验证与确认》。Springer，《人工智能进展》，《计算机科学讲义》，第3809卷：124-127，doi：https://doi.org/10.1007/11589990 137[39] Herten, R. L. M. Chiribiri, A. Breeuwer, M.（2020）《用于心肌灌注磁共振成像定量的物理信息神经网络》。电气工程与系统科学，《图像与视频处理》，doi：arXiv:2011.12844v2[40] Hughes, T. W. Williamson, I. D. Minkov, M. Fan, S.（2019）《波动物理学作为模拟循环神经网络》。《科学进展》，第5卷，第12期，doi：10.1126/sciadv.aay6946[41] Moorkamp, M. Lelievre, P. G. Linde, N. Khan, A.（2016）《地球综合成像：理论与应用》。Wiley，《地球物理学专著系列》，第218卷，第1版：31-40，doi：10.1126/sciadv.aay6946[42] Virieux, J. Asnaashari, A. Brossier, B. Metivier, L. Ribodetti, A. Zhou, W.（2014）《全波形反演引言》。勘探地球物理学家学会，doi：https://doi.org/10.1190/1.9781560803027.entry6[43]Li, ZC., Lin, YZ., Zhang, K. et al. T（2017）《时间域波场重建反演。》。《应用地球物理学》，第14卷：523-528，doi：https://doi-org.manchester.idm.oclc.org/10.1007/s11770-017-0629-6[44] Symes, W. W.（2020）《波场重建反演：一个示例》。AIOP Publishing Ltd，《逆问题》，第36卷，第10期[45] Amestoy, P. 等（2016）《采用并行分块低秩多前沿直接求解器的快速三维频域全波形反演：北海海底电缆数据的应用》。《地球物理学》，第81卷，第6期，doi：10.1190/GEO2016-0052.1[46]Mosely, B. Markham, A. Nissen-Meyer, T.（2020）《利用物理信息深度学习求解波动方程》。https://arxiv.org/abs/2006.11894，doi：https://doi.org/10.1016/j.gsf.2020.07.007

[47] Song, C. Alkhalifah, T. (2021) '基于物理信息神经网络的波场重建反演'。（预印本）doi: arXiv:2104.06897[48]Lipmann, B. A. Schwiner, J. (1950) '散射过程的变分原理'。美国物理学会，第79卷，第4期。doi: https://doi.org/10.1103/PhysRev.79.469[49] Raissi, M. Perdikaris, P. Karniadakis, G. E. (2017) '物理信息神经网络：用于求解涉及非线性偏微分方程的正问题和反问题的深度学习框架'。《计算物理学杂志》，第378卷：686-707。doi: https://doi.org/10.1016/j.jcp.2018.10.045[50]Raissi, M. Perdikaris, P. Karniadakis, G. E. (2017) '物理信息深度学习（第二部分）：非线性偏微分方程的数据驱动发现'。《计算物理学杂志》，第378卷：686-707。doi: https://doi.org/10.1016/j.jcp.2018.10.045[51] Navarro-Jimenez, J. M. Tur, M. Albedla, J. Rodenas, J. J. (2018) '采用浸入边界法的大变形摩擦接触分析'。Springer，《计算力学》，第62卷。doi: 10.1007/s00466-017-1533-x[52] Wunderlich, W. Cramer, H. Steinl, G. (1998) '考虑局部化现象的相关与非相关塑性自适应有限元方法'。《应用力学研究》，第47卷：293-308。doi: https://doi.org/10.1103/PhysRev.79.469[53] Reed, R., Marksll, R. J. (1999) '*Neural Smithing*：前馈人工神经网络中的监督学习'。麻省理工学院出版社[54] Blalock, D. Ortiz, J. J. G. Frankle, J. Guttag, J. (2020) '剪枝的现状是什么？'。《机器学习与系统会议论文集》。doi: arXiv:2003.03033[55] Xenophontos, C. (2003) '使用*Shishkin*网格求解奇异摄动问题时有限元法收敛速度的注记'。爱思唯尔，《应用数学与计算》，第142卷：545-559。doi: 10.1016/S0096-3003(02)00338-7[56]Wang, W. Ma, Y. Zhao, K. Tian, Y. (2020) '机器学习损失函数综述'。Springer，《数据科学年刊》。doi: https://doi.org/10.1007/s40745-020-00253-5[57] Karniadakis, G. E. Kevrekidis, I. G. Lu, L. Perdikaris, P. Wang, S. Yang, L. (2021) '物理信息机器学习'。《自然评论：物理》，第3卷：422-440。doi: https://doi.org/10.1038/s42254-021-00314-5

[58] Baydin, A. G. Pearlmutter, B. A. Radul, A. A. Siskind, J. M. (2018) '机器学习中的自动微分：综述'。《机器学习研究杂志》，第18卷，第1期：1-43[59] Seidel, P. (2011) '数值微分'。麻省理工学院课程讲义，2011年秋季：https://math.mit.edu/classes/18.01/F2011/lecture14.pdf[60] Higham, N. (2002) '数值算法的精度与稳定性'。Siam，第2版：37-38。doi: https://doi.org/10.1137/1.9780898718027[61]Spijker (1971) '有限差分方法误差估计的结构'。《数值数学》，第18卷：73-100。doi: https://doi.org/10.1007/BF01398460[62] Axelsson, O. (1973) '双调和方程数值解的注记'。《IMA应用数学杂志》，第11卷，第2期：213–226。doi: https://doi.org/10.1093/imamat/11.2.213[63] Mustafa, I. Miah, M. Chowdhury, A. Ali, S. Hadi, R. Mehmet, A. A., Chu, Y. (2020) '*Kaup-Kupershmidt*方程的新精确解'。《AIMS数学》，第5卷，第6期：6726-6738。doi: 10.3934/math.2020432[64]Sandstede, B. (2002) '行波稳定性'。爱思唯尔，《动力系统手册》，第2卷：983-1055。doi: https://doi.org/10.1016/S1874-575X(02)80039-X[65]Kalos-Szirmay, L. (2021) '使用对偶数的高阶自动微分'。《Periodica Polytechnica电气工程与计算机科学》，第65卷：1-10。doi: https://doi.org/10.3311/PPee.16341[66] Penunuri, F. Carvente, O. Zambrano, M. A. Peon, R. Cruz-Villar, C. A. (2019) '用于算法微分的对偶数'。《IIngenier´ía》，第23卷，第3期：71-81[67] Ali, K. M. Riaz, F. (2011) '图论在计算机科学中的应用'。第三届计算智能、通信系统与网络国际会议：142-145。doi: 10.1109/CICSyN.2011.40[68] Margossian, C. C. (2018) '自动微分及其高效实现综述'。Wiley，《WIREs数据挖掘与知识发现》，通信系统与网络：142-145。doi: 10.1002/widm.1305[69] Leephakpreeda (2002) '微分方程解的新颖确定方法：通用逼近方法'。爱思唯尔，《计算与应用数学杂志》，第146卷：443-457

[70]Leephakpreeda（1998）《用于求解常微分方程和偏微分方程的人工神经网络》。电气与电子工程师协会，《神经网络汇刊》，第9卷，第5期：987–1000：doi: 10.1109/72.712178

附录 A

A.1 证明与示例

A.1.1 表达式膨胀

人工神经网络从根本上说就是激活函数 $\sigma : \ x \to [-1, 1]$ 这些函数会不断求和并相互嵌套，有时会重复数千次。表达式膨胀是计算中的一种现象，即随着计算的推进，问题的规模呈指数级增长。

考虑以下简单函数

$$f(x) = \tanh\left(\sin\left(\cos\left(x\right)\right)\right) \tag{A.1}$$

它包含 3 个运算。显然，如果它是一个人工神经网络，它将极其简单。对 x 求导，得到

$$f'(x) = -\sin\left(x\right)\cos\left(\cos\left(x\right)\right)\operatorname{sech}^2\left(\sin\left(\cos\left(x\right)\right)\right) \tag{A.2}$$

它包含 9 个运算。我们再次求导，得到

$$\begin{aligned} f''(x) = &-2\sin^2\left(x\right)\cos^2\left(\cos\left(x\right)\right)\operatorname{sech}^2\left(\sin\left(\cos\left(x\right)\right)\right)\tanh\left(\sin\left(\cos\left(x\right)\right)\right) \\ &-\sin^2\left(x\right)\sin\left(\cos\left(x\right)\right)\operatorname{sech}^2\left(\sin\left(\cos\left(x\right)\right)\right) \\ &-\cos\left(x\right)\cos\left(\cos\left(x\right)\right)\operatorname{sech}^2\left(\sin\left(\cos\left(x\right)\right)\right) \end{aligned} \tag{A.3}$$

其中包含 27 次运算。我们再次进行符号微分，得到

$$
\begin{aligned}
f'''(x) = &-\operatorname{sech}^2\left(\sin\left(\cos\left(x\right)\right)\right)\left(4\sin^3\left(x\right)\cos\left(\cos\left(x\right)\right)\right)\sin\left(\cos\left(x\right)\right)\tanh\left(\sin\left(\cos\left(x\right)\right)\right)\\
&+4\cos\left(x\right)\sin\left(x\right)\cos^2\left(\cos\left(x\right)\right)\tanh\left(\sin\left(\cos\left(x\right)\right)\right)\\
&-2\sin^3\left(x\right)\cos^3\left(\cos\left(x\right)\right)\operatorname{sech}^2\left(\sin\left(\cos\left(x\right)\right)\right)\\
&+3\cos\left(x\right)\sin\left(x\right)\sin\left(\cos\left(x\right)\right)-\sin^3\left(x\right)\cos\left(\cos\left(x\right)\right)-\sin\left(x\right)\cos\left(\cos\left(x\right)\right)\\
&-2\sin\left(x\right)\cos\left(\cos\left(x\right)\right)\operatorname{sech}^2\left(\sin\left(\cos\left(x\right)\right)\right)\tanh\left(\sin\left(\cos\left(x\right)\right)\right)\\
&\left(2\sin^2\left(x\right)\cos^2\left(\cos\left(x\right)\right)\tanh\left(\sin\left(\cos\left(x\right)\right)\right)\right)+\sin^2\left(x\right)\sin\left(\cos\left(x\right)\right)\\
&+\cos\left(x\right)\cos\left(\cos\left(x\right)\right)
\end{aligned}
\tag{A.4}
$$

其中包含.....远多于 3 次运算！对于计算机而言，27 次运算并不算多，但大型人工神经网络可能包含数千个全连接节点，因此仅 $f(x)$ 就可能包含数百万次运算。在符号微分过程中，这一规模会急剧增大。

A.1.2 一维波动方程的解

几个世纪以来，人们研究了具有各种初始条件和边界条件的一维波，因此它们非常适合用于测试物理信息神经网络，因为人们对其解已有充分的了解。对于具有狄利克雷边界条件的一维波，可以通过达朗贝尔方法、傅里叶变换或分离变量法求解。下面我们将通过分离变量法，说明（此处插入引用）中所述问题的解为级数解（此处插入引用）。

一维波动方程的形式如下

$$
\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{A.5}
$$

我们寻求一个具有以下形式的解：$u(x,\ t) = X(x)T(t)$，由此得到 s

$$
\frac{1}{X}\frac{d^2X}{dx^2} = \frac{1}{c^2}\frac{1}{T}\frac{d^2}{dt^2} \tag{A.6}
$$

将预期形式代入 A.6。由于两边分别是不同变量的函数，因此它们必须为常数，于是

$$
\begin{aligned}
\frac{1}{X}\frac{d^2X}{dx^2} &= -k^2\\
\frac{1}{c^2}\frac{1}{T}\frac{d^2}{dt^2} &= -k^2
\end{aligned}
\tag{A.7}
$$

然后可以将其作为耦合的常微分方程求解。求得 $X(x)$ 和 $T(t)$ 后可得 s

$$\begin{aligned} X(x) &= A\cos(kx) + B\sin(kx) \\ T(t) &= C\cos(kct) + D\cos(kct) \end{aligned} \tag{A.8}$$

应用边界条件可以求出解中的常数。如果

$$u(x,t) = (A\cos(kx) + B\sin(kx))(C\cos(kct) + D\cos(kct)) \tag{A.9}$$

然后使用 $u(0,\ t) = 0$ 得到 $A = 0$ 和 $k = \frac{n\pi}{2}$，其中 n 是正整数。这给出了一个特解

$$u_n(x,t) = \left(\alpha_n \cos(k_n ct) + \beta_n \cos(k_n ct)\right) \sin\left(\frac{n\pi x}{2}\right) \tag{A.10}$$

初始条件给出 $u_t(x,\ 0) = 0$，因此 $\beta_n = 0$，得到

$$u_n(x,t) = \alpha_n \cos(k_n ct) \sin\left(\frac{n\pi x}{2}\right) \tag{A.11}$$

正弦函数的正交性意味着，对于长度为 2 的弦，有

$$\begin{aligned} \alpha_n &= \int_0^2 u(x,0) \sin\left(\frac{n\pi x}{2}\right) dx \\ &= \int_0^2 x(2-x) \sin\left(\frac{n\pi x}{2}\right) dx \\ &= -\frac{8\pi n \sin(\pi n) + 16\cos(\pi n) - 16}{\pi^3 n^3} \end{aligned} \tag{A.12}$$

其中 $ck_n = \frac{cn\pi}{2} = \frac{n\pi}{2}$（对于 $c = 1$），从而得到级数解

$$u(x,t) = \sum_{n=1}^{\infty} -\frac{8\pi n \sin(\pi n) + 16\cos(\pi n) - 16}{\pi^3 n^3} cos\left(\frac{n\pi t}{2}\right) \sin\left(\frac{n\pi x}{2}\right) \tag{A.13}$$

A.2 误差与绘图代码

所有图表和图的代码都保存在:

`https://github.com/Teddyzander/PhysicsInformedNeuralNetwork`