Advances in Applied Mathematics 应用数学进展, **2024, 13(5), 2380-2391**
Published Online May 2024 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2024.135226

Hans汉斯

Ritz神经网络求解微分方程的数值比较及敏感性分析

史正梅*，刘云美，范小林

贵州师范大学，数学科学学院，贵州 贵阳

收稿日期：2024年4月29日；录用日期：2024年5月22日；发布日期：2024年5月31日

摘 要

Ritz神经网络已经被广泛用来求解微分方程，其中关键一步是近似损失函数中所涉及的定积分，因此求积方法的选取对神经网络逼近解尤为重要。本文通过一些例子进行数值比较分析。首先，我们引入Dirichlet和Neumann边界的边值问题模型。其次，构造神经网络进行模型训练。另外，详细介绍复合梯形求积、复合Simpson求积、三点Gauss求积以及蒙特卡罗求积法。然后，对算例进行数值比较分析，结果表明三点Gauss求积方法更好。最后对神经网络参数敏感性做进一步研究，神经网络的精度随着训练集的增大而提高，当神经元数量达到4 时，再增加神经元数量并不能明显提高精度，而增加隐藏层数量和更换激活函数并没有太大的影响。

关键词

Ritz神经网络，复合梯形求积，复合Simpson求积，三点Gauss求积，蒙特卡罗求积

Numerical Comparison and Sensitivity Analysis of Differential Equations Solved by Ritz Neural Network

Zhengmei Shi*, Yunmei Liu, Xiaolin Fan

School of Mathematics Science, Guizhou Normal University, Guiyang Guizhou

* 通讯作者。

文章引用: 史正梅, 刘云美, 范小林. Ritz神经网络求解微分方程的数值比较及敏感性分析[J]. 应用数学进展, 2024, 13(5): 2380-2391. DOI: 10.12677/aam.2024.135226

Received: Apr. 29th, 2024; accepted: May 22nd, 2024; published: May 31st, 2024

Abstract

Ritz neural networks have been widely used to solve differential equations, in which the key step is to approximate the definite integral involved in the loss function, so the selection of quadrative methods is particularly important for neural networks to approximate the solution. In this paper, some examples are used for numerical comparison. First, we introduce the boundary value problem model of Dirichlet and Neumann boundaries. Secondly, the neural network is constructed to train the model. In addition, the methods of compound trapezoidal quadrature, compound Simpson quadrature, three-point Gauss quadrature and Monte Carlo quadrature are introduced in detail. Then, the numerical comparison and analysis of the examples show that the three-point Gauss quadrature method is better. Finally, the sensitivity of neural network parameters was further studied, and the accuracy of neural network was improved with the increase of training set. When the number of neurons reached 4, increasing the number of neurons could not significantly improve the accuracy, while increasing the number of hidden layers and changing the activation function had no significant impact.

Keywords

Ritz Neural Network, Compound Trapezoid Quadrature, Compound Simpson Quadrature, Three-Point Gauss Quadrature, Monte Carlo Quadrature

1. 引言

随着数据以及计算资源的迅速增长, 人工智能被广泛应用于生物医学 [1, 2],图像处理 [3, 4],语音识别 [5], 油气工程 [6, 7]等领域. 近年来, 应用深度学习来求解微分方程受到越来越多人的关注研究 [8–10].

传统求解微分方程的方法, 如有限元、有限差分等, 需要进行网格划分以及非线性方程组的求解, 求解难度大, 计算成本高, 特别是高维问题, 传统求解方法更加困难.反向传播算法以及自动微分的提出为神经网络求解微分方程奠定了基础, 1998年Lagaris [11]将方程中的初边值条件构造出一个试验解, 得到新的求解微分方程的方法. 后来Raissi [12]提出基于物理信息的神经网络(PINN), 其方法是利用偏微分方程及其边界条件构造残差, 用各项残差之和定义损失函数, 然后拓展到非线性问题并研究连续时间模型和离散时间模型对方程求解与恢复. 鄂维南等 [13]提出了深度里兹的方法(DRM), 通过里兹方法, 构造损失函数, 将原方程的求解转化为变分形式下的优化问题, 其优点是降低求导次数, 减少运算成本, 适合高维偏微分方程求解. H. Sheng 等 [14] 提出一种无惩罚神经网络(PFNN)方法, 可以高效地解决复杂几何上一类二阶边值问题. J. Chen等 [15]采用蒙特卡罗抽样方法而不是蒙特卡罗方法近似损失函数来求解高维偏微分方程, 得出蒙特卡罗抽样方法优于蒙特卡罗方法. J. Taylor 等 [16]基于深度傅里叶残差的方法高效、准确地逼近偏微分方程的解. 以上各种方法的提出基于神经网络具有较好的并行结构、泛化能力、非线性表达功能等, 求解效率高, 自适应强.

然而神经网络在求解微分方程时也存在局限性, 由于损失函数的非凸性, 不能严格保证方法的收敛性 [17], 也可能存在过拟合情况. 在使用神经网络求解微分方程时, 先确定一个求积方法, 然后构造出损失函数的近似函数, 由于被积函数的具体形式未知, 求积方法的选取也不确定, 故存在不同的积分近似方法. 因此, 本文针对复合梯形求积、复合Simpson求积、三点Gauss求积以及蒙特卡罗求积方法进行数值比较分析. 首先, 我们引入了Dirichlet 和Neumann边界的边值问题模型. 其次, 构造神经网络进行模型训练. 另外, 详细介绍了复合梯形求积、复合Simpson求积、三点Gauss求积以及蒙特卡罗求积方法. 然后, 对算例进行数值比较分析, 结果表明三点Gauss求积方法更好. 最后对神经网络参数敏感性做进一步研究, 神经网络的精度随着训练集的增大而提高, 当神经元数量达到4时, 再增加神经元数量并不能明显提高精度, 而增加隐藏层数量和更换激活函数并没有太大的影响.

本文剩余部分安排如下: 在第二节, 我们引入了Dirichlet和Neumann边界的边值问题模型. 在第三节, 我们构造了神经网络进行模型训练. 在第四节, 我们详细介绍了求积方法. 在第五节, 我们对算例进行数值分析. 在第六节, 我们对神经网络参数敏感性进行研究. 在第七节, 我们给出了一些结论性评注.

2. 问题模型

定义边值问题, 其中定义域为$\Omega \subset R^n$,Γ_D和Γ_N分别代表Dirichlet和Neumann 边界条件, $\Gamma_D \cup \Gamma_N = \partial\Omega$,$\Gamma_D \bigcap \Gamma_N = 0$, 其中$n$是单位外法向量, 模型如下:

$$\begin{aligned} G(x, u(x), \nabla u(x), \nabla^2 u(x)) &= f, x \in \Omega, \\ u(x) &= 0, x \in \Gamma_D, \\ \nabla u(x) \cdot n &= g, x \in \Gamma_N . \end{aligned} \tag{1}$$

从Ritz方法 [18]出发, 得到问题的变分形式$J = \int_\Omega A dx + \int_\Omega F dx + \int_{\Gamma_N} B dx$, 其中$B$ 表示方程(1)中与边值条件g 相关的积分, F表示方程(1)中与f相关的积分, A表示方程(1)中与$G(x, u(x), \nabla u(x), \nabla^2 u(x))$相关的积分.

3. 神经网络

由方程(1)的变分形式J来定义神经网络的损失函数$L = \frac{1}{2}\int_\Omega Adx + \int_\Omega Fdx + \int_{\Gamma_N} Bdx$, 将方程(1)求解变为$L$的最小化问题. 我们训练神经网络并输出$U_\theta$, 然后实施Dirichlet边界条件得到方程(1) 的逼近解$U_{nn}(x,\theta)$, 其中θ 是神经网络参数, 表示权重和偏置. 见图 1, 称未实施Dirichlet边界条件前为训练部分, 并定义训练部分的解为U_θ. 实施Dirichlet边界条件后得到神经网络逼近解$U_{nn}(x,\theta)$. 我们定义$\Phi(x)$ 函数, 满足方程(1)的Dirichlet边界条件且值非零 [19], 得到$U_{nn} = \Phi(x)U_\theta$. 将U_{nn}带入损失函数中进行计算, 会涉及到求解导数和积分, 利用自动微分计算相应导数, 不同的求积方法求解积分问题, 下一节将会详细介绍.

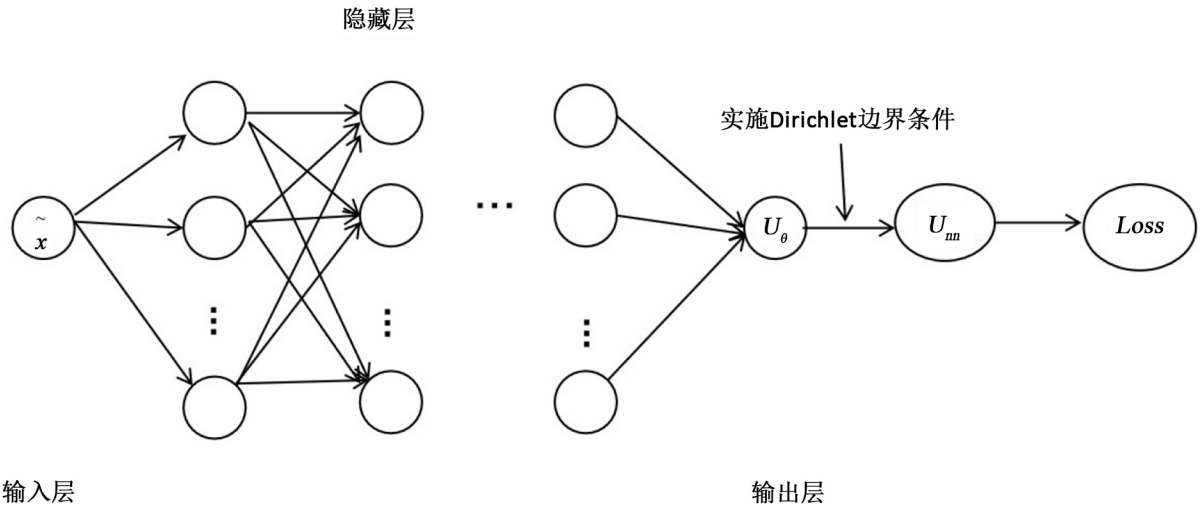

Figure 1. Neural network structure
图 1. 神经网络结构

4. 求积方法

我们用被积函数在节点上函数值的某种线性组合计算积分, 形式为$\int_a^b f(x)dx \approx \sum_{k=0}^n A_k f(x_k)$. 在本文中, 主要关注复合求积、Gauss求积以及蒙特卡罗求积.

4.1. 复合求积公式

令$f(x)$为被积函数, $[a,b]$为被积区间, 将$[a,b]$等分为n个小区间, 小区间长度$h = \frac{b-a}{n}$为步长, 分点记为$x_k = a + kh$ $(k = 0,1,...n)$, 子段$[x_k, x_{k+1}]$的中点为$x_{k+\frac{1}{2}}$, 则复合梯形求积和复合Simpson求

积公式 [20]

$$T_n = \frac{h}{2}(f(a) + 2\sum_{k=1}^{n-1} f(x_k) + f(b))$$
$$S_n = \frac{h}{6}(f(a) + 4\sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}}) + 2\sum_{k=1}^{n-1} f(x_k) + f(b))$$

对求积公式进行误差分析, 令$I = \int_a^b f(x)dx$, $f(x) \in C^2[a,b]$, 则复合梯形求积和复合Simpson求积误差

$$I - T_n \approx -\frac{h^2}{12}\int_b^a f''(x)dx = -\frac{h^2}{12}(f'(b) - f'(a))$$
$$I - S_n \approx -\frac{1}{180}(\frac{h}{2})^4 \int_b^a f^{(4)}(x)dx = -\frac{1}{180}(\frac{h}{2})^4(f^{(3)}(b) - f^{(3)}(a))$$

4.2. Gauss求积公式

Gauss求积公式是定义在区间[-1,1]内, 由节点$x_0, x_1, \ldots x_n$构造的插值型求积公式

$$\int_{-1}^{1} f(x)dx \approx \sum_{k=0}^{n} A_k f(x_k)$$

具有2n+1次代数精度 [20].

如果区间[a,b]是任意的, 则需通过变换 $x = \frac{b-a}{2}t + \frac{a+b}{2}$, $dx = \frac{b-a}{2}dt$, 在[a,b] 上的积分转化为[-1,1]上的积分, 得$\int_a^b f(x)dx \approx \frac{b-a}{2}\int_{-1}^{1} f(\frac{b-a}{2}t + \frac{b+a}{2})dt$.

4.3. 蒙特卡罗求积公式

1998年Caflisch [21]关于蒙特卡罗求积进行了一个综述, 将被积函数积分表示成其在随机点处函数值的数学期望. 考虑积分区域为Ω, 被积函数为f, x 为随机向量, 则有$I = \int_\Omega f(x)dx = E(f(x))$, 其相应的算术平均值为$F = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$, F是积分I 的近似. F是无偏的 [22], 对任何n, 都有$E(F) = I$.

当x为一维区间上的随机变量时, 在区间[a,b], 随机选取n个点, 即$x_i \in [a,b]$, $\forall\, i = 1, ..., n$, 求积公式$\int_a^b f(x)dx \approx \frac{b-a}{n}\sum_{i=1}^{n} f(x_i)$ [23]. 关于误差估计,在α置信水平下, 误差为$|F - I| \le \varepsilon = \frac{\lambda_\alpha \sigma}{\sqrt{n}}$, 蒙特卡罗积分的误差是以$O(\frac{1}{\sqrt{n}})$ 收敛的.

5. 数值实验

Ritz神经网络结合不同的求积方法近似方程的解$u(x)$, 为了对不同求积方法进行数值比较分析, 我们将问题1 和问题2进行数值实验, 借助Pytorch 深度学习框架计算数值问题.

问题1

$$\begin{aligned} -u''(x) &= \sin(x), x \in (0,\pi), \\ u(0) &= 0, \\ u'(\pi) &= -1\ . \end{aligned} \tag{2}$$

此方程的精确解是$u(x) = \sin(x)$.
问题2

$$\begin{aligned} -u''(x) &= -1, x \in (0,10), \\ u(0) &= 0, \\ u'(10) &= 10\ . \end{aligned} \tag{3}$$

此方程的精确解是$u(x) = \frac{x^2}{2}$.

针对方程(2)、(3)先推导变分形式再定义损失函数. 首先将方程(2)、(3)两边同时乘以试探函数$v \in V$, $V = H_0^1(\Omega)$, 由分部积分及边界条件, 得到变分形式$u \in V$, $\forall v \in V$, $J = \int_\Omega u'v'dx - \int_\Omega \sin(x)vdx - \int_{\Gamma_N}(-1)vdx$, $J = \int_\Omega u'v'dx - \int_\Omega(-1)vdx - \int_{\Gamma_N} 10vdx$. 然后定义方程 (2)(??(??19.3 的损失函数$L = \frac{1}{2}\int_\Omega u'v'dx - \int_\Omega \sin(x)vdx - \int_{\Gamma_N}(-1)vdx$, $L = \frac{1}{2}\int_\Omega u'v'dx - \int_\Omega(-1)vdx - \int_{\Gamma_N} 10vdx$.

问题1和问题2使用一个隐藏层含有10 个神经单元、Adam 优化器、Sigmoid 激活函数的神经网络进行训练求解, 对神经网络训练解U_θ 实施Dirichlet边界条件, 令$\Phi(x) = x$, 则神经网络逼近解$U_{nn} = xU_\theta$. 问题1 将区间$[0,\pi]$ 划分为31个等距的小区间, 损失函数为$L = \frac{1}{2}\int_\Omega U'_{nn}U'_{nn}dx - \int_\Omega \sin(x)U_{nn}dx - \int_{\Gamma_N}(-1)U_{nn}dx$. 问题2 将区间$[0,10]$ 划分为15 个等距的小区间, 损失函数为$L = \frac{1}{2}\int_\Omega U'_{nn}U'_{nn}dx - \int_\Omega(-1)U_{nn}dx - \int_{\Gamma_N} 10U_{nn}dx$. 其中近似损失函数处的积分, 采用的求积方法是复合梯形求积、复合Simpson求积、三点Gauss 求积.

Table 1. Problem 1 Relative error of L^2, CPU running time and number of iterations under different quadrature methods

表 1. 问题1 不同求积方法下的L^2相对误差、CPU运行时间及迭代次数

求积方法	$\|u - u_{NN}\|_{L^2}$	时间(s)	迭代次数
复合梯形	2.199175×10^{-2}	183	5054
复合Simpson	2.238300×10^{-4}	288	5100
三点Gauss	1.796842×10^{-4}	102	4126

见表 1和表 2, 展示了问题1和问题2在不同求积方法下的L^2相对误差、CPU运行时间及迭代次数. 我们发现在同样的神经网络以及训练节点下, 三种求积方法在精度、CPU运行时间、迭代次数上各不相同. 明显看出三点Gauss求积方法效率更高, 在较短的运行时间能达到更高的精度, 复合Simpson求积达到相同的精度需要更多的时间, 而复合梯形求积的精度相对较低. 见图 2 和图 3, 展示了问题1 和问题2 使用三点Gauss 求积方法的数值解和逼近解的拟合情况以及相应损失函数的变化. 而蒙特卡罗求积相对于这三种求积方法需要更多的求积节点才能达到较好的精度, 适合高维

Table 2. Problem 2 Relative error of L^2, CPU running time and number of iterations under different quadrature methods

表 2. 问题2 不同求积方法下的L^2相对误差、CPU运行时间及迭代次数

求积方法	$\|u-u_{NN}\|_{L^2}$	时间(s)	迭代次数
复合梯形	1.422728×10^{-2}	124	2932
复合Simpson	5.545139×10^{-5}	172	2008
三点Gauss	3.515455×10^{-5}	89	2279

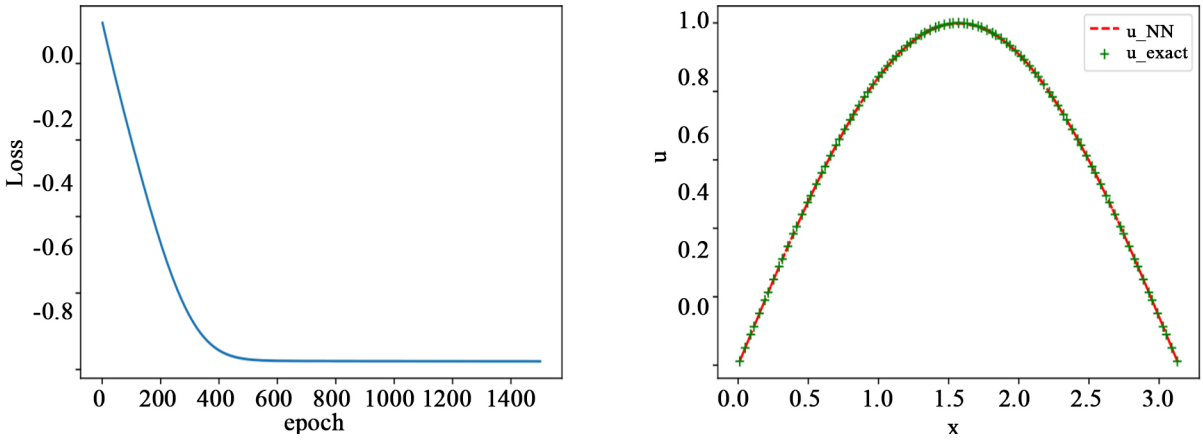

Figure 2. Problem 1 Image of loss function value, numerical solution and approximation solution under three-point Gauss quadrature method

图 2. 问题1 三点Gauss求积方法下损失函数值和数值解、逼近解的图像

积分, 故没有进行具体比较.

6. 神经网络参数敏感性分析

神经网络具有一定的函数逼近能力, 但是神经网络逼近解并不可能完全精确于数值解, 训练集大小和神经网络结构都会对逼近解产生影响. 选取合适的神经元、隐藏层数量以及激活函数十分有必要, 不仅可以达到更佳逼近效果, 还能节省计算成本. 下面将从训练集的大小、神经网络宽度及深度和激活函数等方面分析神经网络精度的变化.

6.1. 训练集大小

搭建一个神经网络模型, 改变训练集大小进行训练, 为表示训练集的大小对神经网络精度的影响, 用L^2 相对误差表示数值解和逼近解之间的接近程度. 先定义L^2 范数$\|f\|_2=(\int_\Omega f^2dx)^{\frac{1}{2}}$,

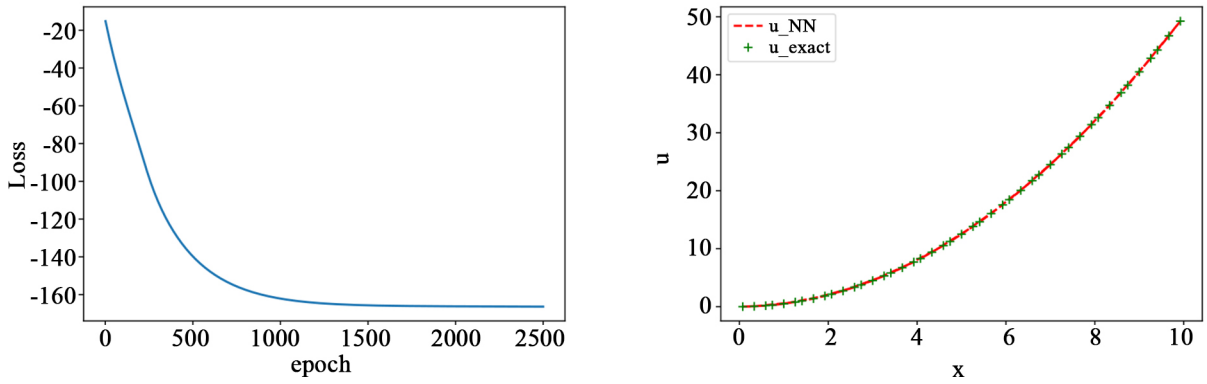

Figure 3. Problem 2 Image of loss function value, numerical solution and approximation solution under three-point Gauss quadrature method

图 3. 问题2三点Gauss求积方法下损失函数值和数值解、逼近解的图像

$f \in C^2(\Omega)$. 可得本文中L^2 相对误差公式

$$L^2\text{相对误差} = \frac{\|U^e - U_{nn}\|_2}{\|U^e\|_2}$$

其中U^e是问题的数值解, U_{nn}是神经网络的逼近解.

对问题1和问题2用蒙特卡罗求积和复合梯形求积方法进行求解, 用一个隐藏层含有10 个神经单元、Adam 优化器、Sigmoid 激活函数的神经网络进行训练. 蒙特卡罗求积下, 模型1和模型2依次分别随机选取100,1000,10000,100000 个点进行数值实验; 复合梯形求积下, 模型1 和模型2依次分别选取32,64,128,256 个点进行数值实验.

Table 3. Relative error of L^2 for different N under monte carlo quadrature

表 3. 蒙特卡罗求积下不同N的L^2相对误差

训练点(N)	$N = 100$	$N = 1000$	$N = 10000$	$N = 100000$
问题1	3.338754×10^{-1}	4.515734×10^{-2}	2.085530×10^{-2}	3.608620×10^{-3}
问题2	1.982314×10^{-1}	3.753440×10^{-2}	1.212293×10^{-2}	2.259647×10^{-3}

Table 4. Relative error of L^2 for different N under compound trapezoid quadrature

表 4. 复合梯形求积下不同N的L^2相对误差

训练点(N)	$N = 32$	$N = 64$	$N = 128$	$N = 256$
问题1	4.220813×10^{-2}	1.212235×10^{-3}	3.093445×10^{-4}	2.199953×10^{-4}
问题2	2.381799×10^{-3}	5.128075×10^{-4}	1.549647×10^{-4}	5.301916×10^{-4}

见表 3和表 4, 展示了在不同训练集下问题1和问题2的L^2相对误差, 随着训练集N的取值越来越大, 该误差逐渐减小. 在这个训练过程中, 训练集的大小可以人为控制, 甚至可以取任意大, 按理论

上来讲, 增加训练集将会提高精度但也增加相应的计算成本, 丢失了神经网络泛化能力, 相比于传统方法的突出优点.

6.2. 神经网络宽度和深度

当增加神经网络的神经元和隐藏层数量时, 可以用来近似更复杂的目标函数, 此时需要训练的参数也在变多, 增加了训练的复杂性. 首先我们固定神经网络为单个隐藏层, 问题1的训练点为32节点, 问题2为16个节点, 观察三种求积方法在不同的神经元下相对误差将如何变化.

Table 5. Problem 1 Relative error of L^2 for different neurons under three quadrature methods

表 5. 问题1三种求积方法下不同神经元的L^2相对误差

神经元(q)	复合梯形	复合Simpson	三点Gauss
$q=1$	2.491040×10^{-1}	2.505580×10^{-1}	2.406169×10^{-1}
$q=2$	1.654949×10^{-1}	1.748590×10^{-3}	1.655599×10^{-3}
$q=4$	1.755143×10^{-2}	1.521407×10^{-4}	2.227213×10^{-4}
$q=8$	1.581488×10^{-2}	1.483948×10^{-4}	1.161781×10^{-4}
$q=16$	2.795062×10^{-2}	2.049432×10^{-4}	1.176479×10^{-4}
$q=32$	1.818506×10^{-2}	3.772021×10^{-4}	2.318206×10^{-4}

Table 6. Problem 2 Relative error of L^2 for different neurons under three quadrature methods

表 6. 问题2三种求积方法下不同神经元的L^2相对误差

神经元(q)	复合梯形	复合Simpson	三点Gauss
$q=1$	1.693368×10^{-2}	1.718070×10^{-2}	1.793181×10^{-2}
$q=2$	1.600680×10^{-2}	4.595825×10^{-3}	4.749406×10^{-3}
$q=4$	1.538911×10^{-3}	6.090653×10^{-5}	7.938960×10^{-5}
$q=8$	1.763098×10^{-3}	3.815646×10^{-5}	3.137327×10^{-5}
$q=16$	1.812437×10^{-3}	4.077426×10^{-5}	3.143023×10^{-5}
$q=32$	1.870956×10^{-3}	8.203752×10^{-5}	2.602972×10^{-5}

见表 5和表 6, 展示了问题1和问题2在不同神经元下L^2相对误差的变化值. 从表 5和表 6中可以看出, 综合三种求积公式和两个问题结果, 一个神经元不足以捕捉函数的复杂性, 在神经元数量较小时, 随着神经元数量慢慢增加, 当数量达到4 时, 数值解和逼近解相对误差达到最佳, 此时再增加神经元的数量并没有明显降低误差.

其次我们固定神经元数量为10, 问题1的训练点为32节点, 问题2为16个节点, 观察三点Gauss求积方法在不同隐藏层下相对误差将如何变化, 其他求积方法相类似.

见表 7, 展示了问题1和问题2在三点Gauss求积方法下, 改变隐藏层数L^2相对误差的变化, 从表中可以发现针对本文问题用深层网络是非必要的, 并没有降低误差, 反而在深层的网络中找到损失函数的局部最小值添加困难, 增加计算成本. 相对一些简单的模型, 使用更复杂网络不一定有多大的

Table 7. Relative errors of L^2 for different hidden layers under three-point Gauss quadrature
表 7. 三点Gauss求积下不同隐藏层的L^2相对误差

隐藏层(K)	$K=1$	$K=2$	$K=3$	$K=4$	$K=6$
问题1	1.796842×10^{-4}	8.260422×10^{-4}	4.171868×10^{-4}	4.407180×10^{-4}	2.198020×10^{-4}
问题2	3.515455×10^{-5}	2.705897×10^{-5}	2.831219×10^{-5}	2.682122×10^{-5}	1.625073×10^{-5}

好处, 较小的神经网络就可以达到所需的精度, 复杂网络反而会带来过拟合的风险. 但对网络结构的进一步研究表明, 深层网络可能减少遇到不好的局部最小值 [24].

6.3. 激活函数

激活函数是构建神经网络的重要参数, 激活函数需要根据研究问题适当选取, 有些激活函数适合, 有些却不适用. 在神经网络的训练过程中, 通过梯度下降优化算法, 找到最佳局部最小值, 需要对损失函数进行求导反向传播来不断训练, 因此通常需要激活函数是连续可导. 本文研究使用的激活函数主要是:Sigmoid、Tanh、SiLU、Softplus及Linear.

Table 8. Problem 1 Relative error of L^2, CPU running time and iteration times of different activation functions under compound Simpson quadrature
表 8. 问题1 复合Simpson求积下不同激活函数的L^2相对误差、CPU运行时间及迭代次数

激活函数	$\|u-u_{NN}\|_{L^2}$	时间(s)	迭代次数
Sigmoid	2.238300×10^{-4}	288	5100
Tanh	1.405257×10^{-4}	277	687
SiLU	2.008837×10^{-4}	285	4126
Softplus	4.274102×10^{-4}	283	5299
Linear	4.018386×10^{-2}	282	656

见表 8, 展示了问题1在不同激活函数下的L^2相对误差、CPU运行时间及迭代次数. 从表**??**中可以得出激活函数在一个隐藏层的神经网络中似乎并没有起太大影响, 只有线性激活函数表现得不乐观, 相对于其他激活函数只能达到10^{-2}次方. 在相差不大的时间内, 达到相同的精度时, Tanh激活函数需要的迭代次数最少, Sigmoid和Softplus激活函数相对较多.

7. 总结

本文基于Ritz神经网络求解微分方程对几种求积方法进行数值比较分析及神经网络参数敏感性研究. 针对问题模型用一些例子对种求积方法进行数值比较分析, 结果表明三点Gauss求积方法表现得更佳, 在较短时间能够达到更好的精度. 对神经网络参数敏感性研究得出, 当训练集增大时神经网络精度也在提高, 神经元数量达到4时, 再增加神经元数量并不能明显提高精度, 而增加隐藏层数量和更换激活函数并没有太大的影响, 相反使用更深的网络可能会造成梯度消失及尽量不使用线性激

活函数. 本文只给了一些一维的数值算例, 但文中的方法还可以推广到二维及以上的区域, 同时也只研究了几种求积方法, 还有更多的求积方法有待探索.

基金项目

贵州师范大学学术新苗基金项目(黔师新苗[2021]A05号).

参考文献

[1] 李金甲, 陈都鑫, 柴人杰, 等. 基于人工智能的蛋白质属性预测的潜能与应用[J]. 药学进展, 2023, 47(10): 733-740.

[2] 荣国光, 等. 医疗保健中的人工智能——综述与预测性案例研究[J]. 工程(英文), 2020, 6(3): 189-211.

[3] 王弈, 李传富. 人工智能方法在医学图像处理中的研究新进展[J]. 中国医学物理学杂志, 2013, 30(3): 4138-4143.

[4] 吴毅, 张小勤. 人工智能在医学图像处理中的研究进展与展望[J]. 第三军医大学学报, 2021, 43(18): 1707-1712.

[5] 马鹏翀. 基于深度学习的语音识别研究[J]. 信息与电脑(理论版), 2021, 33(18): 178-180.

[6] 李道伦, 卢德唐, 孔祥言, 等. BP神经网络隐式法在测井数据处理中的应用[J]. 石油学报, 2007, 28(3): 105-108.

[7] 李道伦, 刘旭亮, 查文舒, 等. 基于卷积神经网络的径向复合油藏自动试井解释方法[J]. 石油勘探与开发, 2020, 47(3): 583-591.

[8] Mai-Duy, N. (2005) Solving High Order Ordinary Differential Equations with Radial Basis Function Networks. *International Journal for Numerical Methods in Engineering*, **62**, 824-852. https://doi.org/10.1002/nme.1220

[9] Ruthotto, L. and Haber, E. (2020) Deep Neural Networks Motivated by Partial Differential Equations. *Journal of Mathematical Imaging and Vision*, **62**, Article 111722. https://doi.org/10.1007/s10851-019-00903-1

[10] McClenny, L.D. and Braga-Neto, U.M. (2023) Self-Adaptive Physics-Informed Neural Networks. *Journal of Computational Physics*, **474**, Article 111722. https://doi.org/10.1016/j.jcp.2022.111722

[11] Lagaris, I.E., Likas, A. and Fotiadis, D.I. (1998) Artificial Neural Networks for Solving Ordinary and Partial Differential Equations. *IEEE Transactions on Neural Networks*, **9**, 987-1000. https://doi.org/10.1109/72.712178

[12] Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2019) 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. https://doi.org/10.1016/j.jcp.2018.10.045

[13] Weinan, E. and Yu, B. (2018) The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Communications in Mathematics and Statistics*, **6**, 1-12. https://doi.org/10.1007/s40304-018-0127-z

[14] Sheng, H. and Yang, C. (2021) PFNN: A Penalty-Free Neural Network Method for Solving a Class of Second-Order Boundary-Value Problems on Complex Geometries. *Journal of Computational Physics*, **428**, Article 110085. https://doi.org/10.1016/j.jcp.2020.110085

[15] Chen, J., Du, R., Li, P. and Lyu, L. (2021) Quasi-Monte Carlo Sampling for Solving Partial Differential Equations by Deep Neural Networks. *Numerical Mathematics*, **14**, 377-404. https://doi.org/10.4208/nmtma.OA-2020-0062

[16] Taylor, J.M., Pardo, D. and Muga, I. (2023) A Deep Fourier Residual Method for Solving PDEs Using Neural Networks. *Computer Methods in Applied Mechanics and Engineering*, **405**, Article 115850. https://doi.org/10.1016/j.cma.2022.115850

[17] Mishra, S. and Molinaro, R. (2023) Estimates on the Generalization Error of Physics-Informed Neural Networks for Approximating PDEs. *IMA Journal of Numerical Analysis*, **43**, 1-43. https://doi.org/10.1093/imanum/drab093

[18] 李荣华. 偏微分方程数值解法(第二版) [M]. 北京: 高等教育出版社, 2010: 11.

[19] Rivera, J.A., Taylor, J.M., Pardo, D., *et al.* (2022) On Quadrature Rules for Solving Partial Differential Equations Using Neural Networks. *Computer Methods in Applied Mechanics and Engineering*, **393**, Article 114710. https://doi.org/10.1016/j.cma.2022.114710

[20] 杨一都. 数值计算方法[M]. 北京: 高等教育出版社, 2008: 4.

[21] Caflisch, R.E. (1998) Monte Carlo and Quasi-Monte Carlo Methods. *Acta Numerica*, **7**, 1-49. https://doi.org/10.1017/S0962492900002804

[22] 雷桂媛. 关于蒙特卡罗及拟蒙特卡罗方法的若干研究[D]: [博士学位论文]. 杭州: 浙江大学, 2003.

[23] 张艳. 利用蒙特卡罗方法求解数值积分[J]. 高等数学研究, 2023, 26(1): 44-46+61.

[24] Remi Hernandez. 神经网络算法求解微分方程的参数敏感性研究[D]: [硕士学位论文]. 厦门: 厦门大学, 2020.