Statistics and Application 统计学与应用, 2020, 9(6), 1048-1059
Published Online December 2020 in Hans. http://www.hanspub.org/journal/sa
https://doi.org/10.12677/sa.2020.96110

Hans汉斯

前馈神经网络在多元函数逼近中的应用

葛悠然[1]，翟九媛[2]，马尧鹏[3]，王汉权[1*]

[1]云南财经大学，统计与数学学院，云南 昆明
[2]云南大学，经济学院，云南 昆明
[3]云南财经大学，云南 昆明
Email: u__u95@126.com, zhai_jiuyuan@163.com, 2757574611@qq.com, *wanghanq@cz3.nus.edu.sg

收稿日期：2020年11月29日；录用日期：2020年12月22日；发布日期：2020年12月31日

摘 要

给定一组数据(例如一些点及相应点处的函数值)，找到未知函数的表达公式——函数逼近问题是数学与工程应用中的一个基本问题。传统的数值方法多采用多项式插值法(例如拉格朗日插值法、牛顿插值法、三次样条法等)，本文通过构造前馈神经网络函数得到未知函数的表达式，讨论其处理函数逼近问题的优缺点。具体说来，先介绍训练多元函数的前馈神经网络的详细计算过程，然后分析隐含层节点数目对该网络的精度影响问题。最后通过数值计算结果证实前馈神经网络可用来逼近一元函数、二元函数、三元函数，能够达到较高的计算精度。本文的讨论适用于其他类人工神经网络在四元或四元以上的多元函数逼近问题的研究，也有助于理解相关人工神经网络的基本性质与作用。

关键词

前馈神经网络，函数逼近，隐含层节点数目

Application of Feedforward Neura Networks in Multivariate Function Approximation

Youran Ge[1], Jiuyuan Zhai[2], Yaopeng Ma[3], Hanquan Wang[1*]

[1]School of Statistics and Mathematics, Yunnan University of Finance and Economics, Yunnan Kunming
[2]School of Economics, Yunnan University, Yunnan Kunming
[3]Yunnan University of Finance and Economics, Yunnan Kunming
Email: u__u95@126.com, zhai_jiuyuan@163.com, 2757574611@qq.com, *wanghanq@cz3.nus.edu.sg

Received: Nov. 29th, 2020; accepted: Dec. 22nd, 2020; published: Dec. 31st, 2020

*通讯作者。

文章引用: 葛悠然, 翟九媛, 马尧鹏, 王汉权. 前馈神经网络在多元函数逼近中的应用[J]. 统计学与应用, 2020, 9(6): 1048-1059. DOI: 10.12677/sa.2020.96110

Abstract

Given a set of data (such as some points and function values at corresponding points), finding the expression formula of the unknown function—the function approximation problem is a fundamental problem in mathematics and engineering applications. Traditional numerical methods mostly use polynomial interpolation (such as Lagrangian interpolation, Newton interpolation, cubic spline method, etc.). In this paper, the expression of the unknown function is obtained by constructing the feedforward neural network function, and the advantages and disadvantages of the approximation problem of the processing function are discussed. Specifically, the detailed calculation process of the feedforward neural network for training multivariate functions is first introduced, and then the influence of the number of hidden layer nodes on the accuracy of the network is analyzed. Finally, the numerical results show that the feedforward neural network can be used to approximate the unary function, the binary function and the ternary function, which can achieve higher calculation accuracy. The discussion in this paper is applicable to the study of multivariate function approximation problems of other artificial neural networks in quaternary or quadruple, and it also helps to understand the basic properties and effects of related artificial neural networks.

Keywords

Feedforward Neural Network, Function Approximation, Number of Hidden Layer Nodes

Copyright © 2020 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

给定一组数据，找到能代表该数组的函数表达式是数学与工程应用中的一个基本问题。函数逼近问题能够应用于解决地震勘探、信号处理、物理探矿等方面的实际问题[1]。随着计算机技术的不断发展，人工神经网络已经成为简便的计算工具。人工神经网络处理数据具有计算简便、耗时短、精度高等优点[2]，很多大数据复杂计算问题应用人工神经网络来解决。前馈神经网络作为人工神经网络的一个重要分支，研究它在多元函数逼近中的应用，一方面可解决前述基本问题，另一方面，通过研究该基本问题可验证它的性能、掌握训练神经网络的主要步骤。

由于人工神经网络解决函数逼近问题时便捷、精准的特点，在上世纪，人们就开始了对它的研究。早在 20 世纪 90 年代，国外的学者们就已经开始了对人工神经网络解决函数逼近问题的早期探索。在 1993 年，Bulsari A. [3]提出了利用前馈神经网络解决一些特殊函数的逼近问题，并选用 Sigmoid 函数作为网络的传递函数来构造函数逼近网络。在 1998 年，Suzuki 和 Shin [4]证明了三层人工神经网络可以应用于三角函数、分段线性函数的逼近问题。同年，Twomey J. M.和 Smith A. E. [5]应用人工神经网络可以解决数据不足时的函数逼近问题。随后，在 2005 年，Ferrari S.和 Stengel R. F. [6]利用前馈神经网络解决非线性函数逼近问题。在 2013 年，S. Yao、C. J. Wei 和 Z. Y. He [7]将前馈神经网络中的 RBF 网络、BP 网络和 GRNN 网络的性能进行比较，分析得出 RBF 网络在函数逼近中的精度和速度方面最优。

国内学者在 20 世纪末开始对人工神经网络解决函数逼近的问题进行研究。在 1997 年，韦岗、李华和徐秉铮[8]首次证明了前馈神经网络的隐含层神经元数目足够多时，其多维函数逼近能力与维数无关。

该定理大大简化了前馈多层神经网络函数逼近问题的分析难度。在 2005 年，王强、余岳峰和张浩炯[9]以一个一维非线性函数为例论述了一个单隐层的前馈神经网络解决函数逼近问题的过程。随后，在 2009 年，侯木舟[10]提出了前馈多层神经网络函数逼近问题在股市数据预测、环境数据预测、EGG 信号预测等方面的应用问题。在 2016 年，李鹏柱[11]研究了前馈神经网络的函数逼近中优化激活函数和固定权值的问题。

本文的研究建立在前人的研究基础上，主要有下述贡献点：寻找前馈神经网络中最适隐含层节点数目来增加网络的逼近精度；提出一种新的训练方法——贝叶斯归一化法[12]，将参数映射到小范围内，使得求解极小值问题的收敛速度更快捷；最后总结了所构造的网络在所有函数逼近问题中的普遍应用的可能性。

2. 理论准备

2.1. 前馈神经网络

人工神经网络可分为前馈型和反馈型，相对于反馈型神经网络，前馈型神经网络各网络层的神经元间互不连接[13]，即此网络各层之间没有反馈，是一种最简单的神经网络。

前馈神经网络中一般包括多层网络，每层网络有多个神经元。信息输入到输入层，单向传递到隐含层加入权值和阈值，经传递函数运算后到输出层，最后由输出层输出结果[14]。

如图 1 所示，记 $w^{(0)}$ 为第 1 层隐含层的权值矩阵，$w^{(l)}$ 为第 2 层隐含层的权值矩阵，$w^{(l-1)}$ 为第 l 层隐含层的权值矩阵，$w^{(l)}$ 为输出层的权值矩阵；记 $b^{(0)}$ 为第 1 层隐含层的阈值矩阵，$b^{(l)}$ 为第 2 层隐含层的阈值矩阵，$b^{(l-1)}$ 为第 l 层隐含层的阈值矩阵，$b^{(l)}$ 为输出层的阈值矩阵；$f^{(0)}(\cdot)$ 为第 1 层隐含层的传递函数，$f^{(1)}(\cdot)$ 为第 2 层隐含层的传递函数，$f^{(l-1)}(\cdot)$ 为第 l 层隐含层的传递函数，且所有传递函数默认为光滑连续的函数；$f^{(l)}(\cdot)$ 为输出层函数[6] [15]。

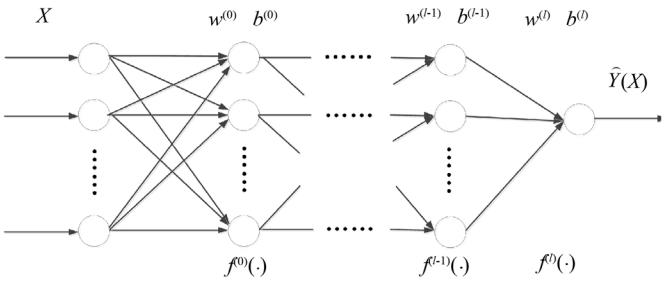

Figure 1. Feed forward neural network diagram
图 1. 前馈神经网络简图

令 x 为输入数据矩阵，计算该数据传递到第 1 层隐含层得到输出值为

$$z^{(1)} = f^{(0)}\left(w^{(0)} * x + b^{(0)}\right) \tag{1}$$

然后经过第 2 层隐含层得到输出值为

$$z^{(2)} = f^{(1)}\left(w^{(1)} * z^{(1)} + b^{(1)}\right) \tag{2}$$

依次向下传递到第 l 层隐含层得到输出值为

$$z^{(l)} = f^{(l-1)}\left(w^{(l-1)} * z^{(l-1)} + b^{(l-1)}\right) \tag{3}$$

最后传递到输出层输出网络的结果为

$$y_{w,b}(x)=f^{(l)}\left(w^{(l)}*z^{(l)}+b^{(l)}\right) \tag{4}$$

前馈神经网络的工作过程分为正向传递学习和反向传递训练两个过程[16],其工作原理主要是对所涉及到的参数进行调整，最终使得网络误差最小。其中正向传递学习过程中，数据通过输入层进入网络，在隐含层中，代入权值和阈值、传递函数运算后，传到输出层。比较输出值和期望值，如若差别很大，则进行反向传递训练过程。在反向传递训练过程中，采用特定的训练方式，不断更新网络的权值和阈值，直到网络的输出与期望输出之间误差最小，最终得到一个精准的前馈神经网络。

2.2. 函数逼近问题

在数学研究和工程应用中存在以下函数逼近问题：已知函数 $F(x)$，找到某类选定函数中的特定函数 $g(x)$，使得 $g(x)$ 可以在一定意义下近似表示 $F(x)$，并求出近似误差。在函数逼近问题中，逼近已知函数的函数类是多样化的；即使确定了函数类范围，其中特定逼近函数 $g(x)$ 仍然是多种多样的；另外特定逼近函数 $g(x)$ 与被逼进函数 $F(x)$ 的逼近误差的定义方式也不同。逼近函数有许多方法，包括插值法、线性(或非线性)回归法、基函数展开法、数理统计方法[16]。

本文通过构造前馈神经网络函数来讨论函数逼近问题。它可以简单理解为用输入数据多次训练网络逼近函数，从而得到一个精准的神经网络函数，以便后续调用和改进。

2.3. 构造函数的逼近网格

构造逼近函数的前馈神经网络大致分为以下四个步骤：

第一步，给定的一组数据。本文中选取函数定义域中的 N 个点及点处的函数值作为已知数据：$\left(x_i,F\left(x_i\right)\right)\left(i=1,2,\cdots,N\right)$。

第二步，选择前馈神经网络的结构，包括网络层数目，各层神经元的数目，以及传递函数，进而构造出前馈神经网络函数(参见公式(4))。

第三步，训练前馈神经网络的参数。

若前馈神经网络的误差函数[17]定义为

$$f(w,b)=\frac{1}{N}\sum_{i=1}^{N}\left[y_{w,b}(x_i)-F(x_i)\right]^2 \tag{5}$$

其中，$y_{w,b}(x)$ 为神经网络输出函数，其形式如公式(4)所示。$\left(x_i,F\left(x_i\right)\right)\left(i=1,2,\cdots,N\right)$ 为给定的 N 个样本，w 为权值矩阵，b 为阈值矩阵。

该训练过程的目标是求出特殊的权值矩阵 w^* 和阈值矩阵 b^*，使得此时的目标误差函数 $f\left(w^*,b^*\right)$ 最小(接近于 0)，输出此时的权值矩阵 w^* 和阈值矩阵 b^*，代入激活函数中求得该神经网络函数。训练过程就是利用迭代法求解如下极小值问题：

找特殊的权值矩阵 w^* 和阈值矩阵 b^* 使得 $f\left(w^*,b^*\right)\le f(w,b)$ 对任意的权值矩阵 w 和阈值矩阵 b 都成立。

第四步，得出该网络函数表达式 $y_{w^*,b^*}(x)$，并画出所训练的前馈神经网络和原函数的对比图，同时结合网络的输出值的绝对值误差图来分析网络的精度。

注意：

(1) 在第二步中，隐含层节点数目多少与网络的逼近精度有关。对于隐含层节点数目，过少会导致网络不拟合函数，而节点数目太多会造成网络的过适性现象，所以可以通过改变其隐含层节点数目来优化

网络的效果。本文中各隐含层的传递函数选择“ $logsig(x)$ ”函数[5]，此函数光滑连续符合网络的传递函数要求，其函数表达式如下：

$$logsig(x)=\frac{1}{1+e^{-x}} \tag{6}$$

在隐含层和输出层之间的传递函数选择“ $purelin(x)$ ”函数[5]，此函数值可以取任意值，符合网络的输出范围要求。其函数表达式如下：

$$y=x \tag{7}$$

(2) 在第三步中，选择何种求极小值问题的迭代方法也很重要，这会影响整个网络的逼近精度。本文选择 MATLAB 中的“ $trainbr(x)$ ”函数[5]来训练该网络。该函数利用一种随机梯度下降法——贝叶斯归一化法来求解上述极小值问题。在贝叶斯归一化法使用过程中[17]，假定网络的权值和阈值是按照指定分布随机可变的，根据贝叶斯法则更新权值和阈值密度函数，再求均方误差函数最小时的权值和阈值。“ $trainbr(x)$ ”函数将参数映射到小范围内，使得求解极小值问题的收敛速度更快捷。

3. 构造逼近多元函数的前馈神经网络

3.1. 一元函数

对于一元函数的逼近网络，我们选择“1-*n*-1”类型的前馈神经网络，即该网络含有一个输入层、一个输出层和一个隐含层，其中输入层和输出层各有一个神经元，隐含层有 *n* 个神经元。通过试验发现不同函数的最适隐含层神经元数目不同，所以要具体函数具体分析。

以函数 $F(x)=\begin{cases}|x|*\cos(x), x<0\\ e^{x}-1, x\geq 0\end{cases}$ 为例，定义 $\vec{x}=(x)$，选取 $x\in[-2\pi,2\pi]$ 时、间距为 0.1 的 $(\vec{x},F(\vec{x}))$ 数据组作为输入数据，构建的前馈神经网络参数设定为训练迭代次数为 10^5 次，训练目标误差函数值小于 1×10^{-10}，当隐含层节点数目为 27~31 个时，网络对该一元函数逼近的绝对值误差($|F(\vec{x})-y_{w,b}(\vec{x})|$)如表 1 所示：

Table 1. Absolute value error ($|F(\vec{x})-y_{w,b}(\vec{x})|$) of $F(x)=\begin{cases}|x|*\cos(x), x<0\\ e^{x}-1, x\geq 0\end{cases}$ neurons in different hidden layers

表 1. $F(x)=\begin{cases}|x|*\cos(x), x<0\\ e^{x}-1, x\geq 0\end{cases}$ 函数不同隐含层神经元的绝对值误差($|F(\vec{x})-y_{w,b}(\vec{x})|$)

隐含层节点数目/个	训练次数/次	绝对值误差 $\lvert F(\vec{x})-y_{w,b}(\vec{x})\rvert$
27	262	9.97998e−06
28	229	9.96494e−06
29	137	9.87421e−06
30	217	9.94987e−06
31	914	0.00001

表 1 结合不同隐含层节点数目的网络逼近函数图观察得，节点数目在 27~29 时，网络的误差逐渐变小，节点数目在 30 及以上时，网络的误差变大。综上可得，隐含层节点数目为 29 个时，对应前馈神经网络对函数 $F(x)=\begin{cases}|x|*\cos(x), x<0\\ e^{x}-1, x\geq 0\end{cases}$ 逼近效果最好。

加入以上最适隐含层神经元数目后，创建训练该网络，然后输出训练后网络的权值矩阵和阈值矩阵如下：

$$iw1=\begin{bmatrix}-17.5017\\17.7843\\-17.2097\\-18.1494\\18.0743\\-17.8794\\17.5053\\-18.3826\\17.9985\\17.9110\\-18.3157\\19.1000\\-17.8663\\17.9186\\-18.9242\\-17.9863\\18.4468\\-17.6651\\19.0108\\22.9044\\-22.9604\\-23.1927\\-20.8928\\-18.8652\\-13.8623\\-24.4628\\29.8524\\-25.2123\\-24.2074\\22.9180\\16.4425\end{bmatrix},\quad b1=\begin{bmatrix}17.7830\\-16.3904\\15.7378\\14.5076\\-13.1672\\11.5042\\-10.6494\\9.6780\\-8.2057\\-6.6430\\5.7357\\-4.6570\\2.9741\\-1.9774\\0.2439\\-0.1555\\0.6655\\-1.8936\\4.1492\\6.3582\\-7.6563\\-10.9059\\-11.0087\\-7.5742\\-9.8597\\-18.5480\\23.8604\\-21.3395\\-21.9026\\22.2410\\18.7242\end{bmatrix}$$

$$lw2=\begin{bmatrix}-0.2707 & 0.4543 & 0.4056 & 0.1569 & -0.1736 & 0.1094 & 0.1429 & -0.1671 & 0.1526 & 0.0017 & 0.1467 & -0.1536\\ 0.1332 & 0.1195 & -0.5946 & 0.7799 & 0.4650 & -0.0359 & -0.1729 & -0.1301 & 0.1556 & 0.0987 & 0.1081 & 0.4143\\ -1.2026 & -0.1712 & 0.1004 & -0.3410 & -0.3413 & 0.2856 & -0.9539\end{bmatrix}$$

$$b2=0.3253$$

由上面计算得到的数值，可得如下神经网络函数：

$$y_{w,b}(x)=pureline\left(iw2*logsig\left(iw1*x+b1\right)+b2\right) \tag{8}$$

从图 2 中(右)图可以看出逼近该一元函数的前馈神经网络的误差绝对值在$\left(0,8\times10^{-5}\right)$之间，结合其中

网络输出结果与输入样本对比图分析可知，该网络逼近效果较好。

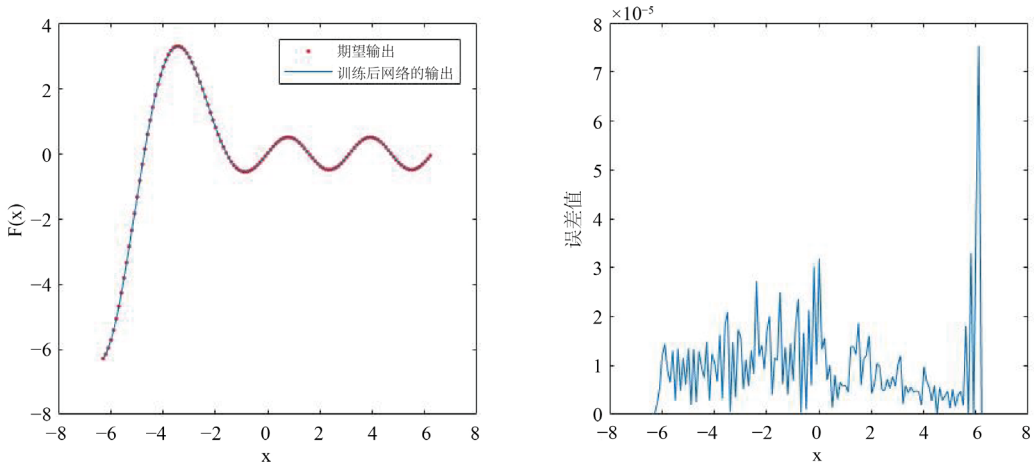

Figure 2. (left) A comparison between the output of the network and the input sample, and (right) an absolute value error ($\left|F(x)-y_{w,b}(x)\right|$) graph approximated by the network

图 2. (左) 该网络输出结果与输入样本的对比图，(右) 该网络逼近的绝对值误差($\left|F(x)-y_{w,b}(x)\right|$)图

3.2. 二元函数

对于二元函数的逼近网络，我们选择“1-*n*-*m*-1”类型的前馈神经网络，即该网络含有一个输入层、一个输出层和两个隐含层，其中输入层和输出层各有一个神经元，两个隐含层分别有 *n* 个和 *m* 个神经元。

以函数 $F(x_1,x_2)=e^{x_1}*\cos(x_2)+2$ 为例，定义 $\vec{x}=(x_1,x_2)$，选取 $(x_1,x_2)\in[-10,10]^2$ 时、间距为 0.1 的 $(\vec{x},F(\vec{x}))$ 数据组作为输入数据，构建的前馈神经网络参数设定为训练迭代次数为 10^5 次，训练目标误差函数值小于 1×10^{-10}，当隐含层节点数目分别为 15~19 个时，网络对 1×10^{-10} 该函数逼近误差如表 2 所示：

Table 2. Absolute value error ($\left|F(\vec{x})-y_{w,b}(\vec{x})\right|$) of neurons in different hidden layers of this binary function

表 2. 该二元函数不同隐含层神经元的绝对值误差($\left|F(\vec{x})-y_{w,b}(\vec{x})\right|$)

第 1 层隐含层节点数目/个 绝对值误差 $\left\|F(\vec{x})-y_{w,b}(\vec{x})\right\|$ 第 2 层隐含层节点数目/个	15	16	17	18	19
15	1.76e−04	1.11e−05	5.28e−05	1.04e−05	1.14e−04
16	1.71e−05	9.95e−04	9.95e−06	9.99e−06	1.05e−05
17	1.03e−05	3.60e−05	1.13e−05	1.38e−05	9.98e−06
18	1.40e−05	1.76e−05	9.98e−06	9.98e−06	9.98e−04
19	1.24e−05	1.67e−05	0.000013	9.97e−06	0.00001

分析表 2 数据，结合不同隐含层节点数目的网络逼近函数图观察得，第 1 层隐含层神经元数目为 17 且第 2 层隐含层神经元数目为 16 时，该网络的精度最高。

加入以上最适隐含层神经元数目后，创建训练该网络，然后输出训练后网络的权值矩阵和阈值如下：

$$iw1=\begin{bmatrix}-4.5422 & 0.5258\\ -0.7558 & 2.5546\\ 1.2497 & -1.0693\\ -2.1801 & -1.3271\\ 0.9576 & 2.4907\\ 0.5849 & 0.9661\\ -1.2717 & 1.8328\\ -0.9540 & -2.2704\\ -0.4145 & -1.9829\\ -1.5737 & -1.2474\\ 1.3511 & 1.5302\\ -2.7319 & 0.4415\\ -1.5054 & 2.3729\\ -1.8429 & 2.4382\\ -1.9307 & 0.1799\\ -0.5447 & 2.5053\\ -0.5863 & 2.9935\end{bmatrix},\quad b1=\begin{bmatrix}6.5451\\ -2.5071\\ -1.1402\\ 2.9001\\ -3.5693\\ -2.3158\\ 0.9891\\ -2.1784\\ -0.3553\\ 0.3076\\ -0.0247\\ 3.1840\\ -0.4508\\ -1.8328\\ 1.0981\\ 2.5750\\ -4.1249\end{bmatrix},\quad b2=\begin{bmatrix}0.3778\\ 0.2395\\ -0.8920\\ -0.9407\\ -0.4885\\ 0.2707\\ 0.0726\\ -0.5127\\ -0.6129\\ 0.3651\\ 0.8171\\ 0.1503\\ -0.2486\\ -0.9560\\ -0.4690\\ 0.4579\end{bmatrix}$$

$$lw2=\begin{bmatrix}0.0755 & 0.1292 & 0.2770 & 0.9822 & -0.5831 & 0.2791 & 0.0302 & -0.4355 & 0.2783 & -0.2835 & 0.0448\\ -0.7550 & 0.0882 & 0.2814 & -0.3348 & -0.0985 & 0.0813 & -0.8132 & 0.0239 & 0.4978 & -0.4246 & -0.1212\\ 0.1976 & 0.0167 & 0.0396 & 0.1550 & 0.2899 & -0.1324 & -0.3160 & 0.1327 & -0.0497 & -0.3458 & -0.1618\\ 0.3514 & 0.2381 & 1.4277 & -2.3039 & 1.7779 & -2.7537 & 1.0981 & -0.2338 & 3.8810 & -0.7455 & 1.0516\\ -1.8539 & 0.8102 & 0.4042 & 1.1222 & -0.2190 & 3.3436 & 2.3855 & 0.5285 & -0.6148 & -0.8265 & 0.0996\\ 0.2363 & -0.1678 & -0.3108 & 1.5485 & -0.0751 & 0.3754 & -0.7284 & 2.6631 & 0.8261 & -0.0250 & 1.0486\\ -0.5605 & -0.2044 & 1.7929 & 2.0254 & -2.2513 & 2.5820 & -5.3369 & 0.6247 & 2.3541 & 4.6387 & -1.1494\\ 2.0478 & -2.6739 & -1.0560 & 1.6767 & 1.1929 & 0.9182 & 2.4935 & 2.0634 & -0.5805 & 0.0749 & 2.1404\\ -1.2802 & 0.3370 & 0.0311 & -1.1869 & -0.6591 & 1.0518 & -0.5673 & 1.6187 & -2.0341 & -0.2454 & -0.3587\\ -0.2745 & -4.4865 & -0.0955 & 1.7719 & -0.8895 & 0.4570 & -0.9613 & 1.8746 & -0.7379 & -0.4048 & -1.2179\\ 0.6990 & -1.2807 & 0.4181 & -0.5899 & -0.3712 & -0.0891 & -0.4858 & -1.1889 & -1.9462 & 1.7427 & -0.6592\\ -1.0284 & -0.0216 & -0.1151 & -0.8242 & 0.5599 & -1.1087 & 0.5822 & 0.3767 & -0.0022 & 0.9677 & -0.0527\\ -0.0454 & 1.7684 & -0.5127 & -0.9613 & 3.7704 & 1.4942 & -0.8043 & 0.7199 & 4.4897 & 0.0787 & 2.2926\\ 1.7007 & 2.9853 & 1.6885 & -0.8633 & 2.2445 & 2.4128 & 0.9374 & 0.8489 & -4.4151 & 1.4585 & 0.0740\\ -0.0063 & 0.3401 & -1.1314 & 0.0090 & 0.9886 & -0.4831 & -0.7750 & -0.7132 & -0.2355 & 1.1755 & 3.7296\\ 0.3776 & -0.1321 & 0.1114 & 2.1160 & 0.2985 & 0.1333 & -0.3531 & 1.2411 & 0.2862 & 0.4549 & 0.5579\\ 0.6507 & 1.5492 & 0.4536 & 0.1173 & -0.6705 & 0.5543 & -0.4632 & -0.6124 & -0.4840 & 1.0591 & -0.6915\\ 1.8201 & 1.3026 & -1.0832 & 0.9080 & 3.3683 & 0.1816 & 0.9367 & 1.6836 & 1.1626 & -0.0348 & -1.0529\\ 1.7858 & 0.7038 & 0.6100 & 1.2945 & -3.1167 & 0.8436 & 0.5827 & 0.6221 & -0.1458 & -0.7173 & 0.8186\\ 0.4904 & -0.0935 & 0.1625 & -0.0451 & -0.9619 & 0.0553 & 1.1837 & 0.0834 & 0.3208 & -0.2315 & -0.0420\\ 0.4551 & -1.2312 & -0.6695 & 0.3441 & -1.3419 & 1.3449 & -0.2517 & -0.5571 & -0.3975 & -0.3913 & -0.5093\\ -0.0344 & -0.0146 & -0.5689 & -0.7353 & -0.3980 & -0.8817 & -0.6159 & 0.5502 & -0.5449 & -0.0720 & -0.4644\\ -0.0052 & -0.2850 & 0.2373 & 0.3318 & 0.6767 & 0.8991 & -0.4579 & 0.9276 & 0.1970 & -0.7285 & 0.3810\\ -0.4564 & 0.0071 & -1.1508 & 0.0985 & 0.7294 & -0.5010 & -0.4649 & 0.5116 & -0.3485 & 0.5007 & -0.4760\\ 0.3020 & 0.7285 & -0.7886 & -0.0183 & -0.1053 & 1.1088 & 0.5971 & 0.3443 & & & \end{bmatrix}$$

$$b3=1.8061$$

由上面计算得到的数值，可得如下神经网络函数：

$$y_{w,b}(\vec{x}) = pureline\left(logsig\left(iw2 * logsig\left(iw1 * \vec{x} + b1\right) + b2\right) + b3\right) \tag{9}$$

从图 3 中(右)图可以看出逼近该二元函数的前馈神经网络的误差绝对值在$\left(0,1\times10^{-4}\right)$之间，结合其中网络原函数图像与网络输出结果图对比分析可知，该网络逼近效果较好。

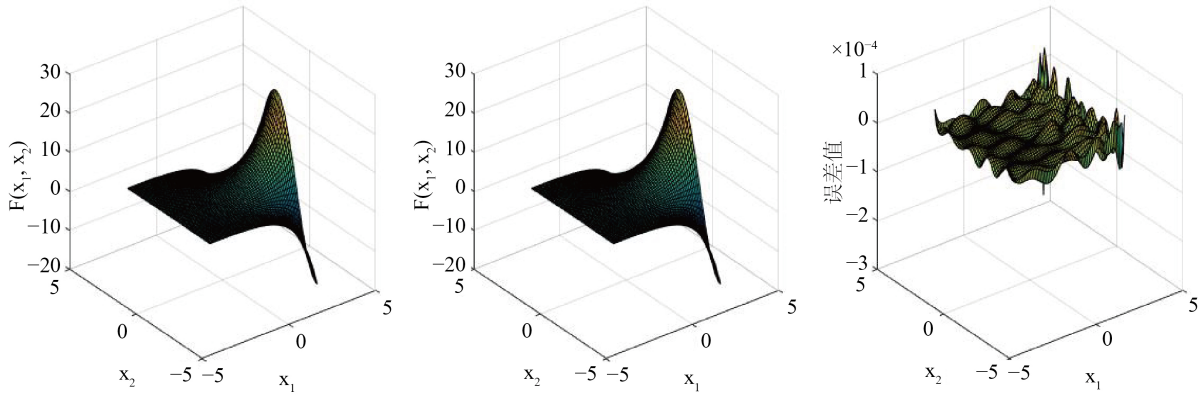

Figure 3. (left) original function image, (middle) output result diagram, (right) absolute value error ($\left|F(\vec{x}) - y_{w,b}(\vec{x})\right|$) diagram of the network approximation

图 3. (左) 原函数图像，(中) 该网络输出结果图，(右) 该网络逼近的绝对值误差($\left|F(\vec{x}) - y_{w,b}(\vec{x})\right|$)图

3.3. 三元函数

对于三元函数的逼近网络，我们选择“1-*n*-*m*-*k*-1”类型的前馈神经网络，即该网络含有一个输入层、一个输出层和三个隐含层，其中输入层和输出层各有一个神经元，三个隐含层分别有 *n* 个、*m* 个和 *k* 个神经元。

以函数 $F(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 1$ 为例，定义 $\vec{x} = (x_1, x_2, x_3)$，选取 $x_1, x_2, x_3 \in [-3,3]^3$ 时、间距为 0.3 的 $(\vec{x}, F(\vec{x}))$ 数据组作为输入数据，构建的前馈神经网络参数设定为训练迭代次数为 10^5 次，训练目标的误差小于 1×10^{-10}，当隐含层节点数目分别为 8~12 个时，网络对该函数逼近误差如表 3~表 5 所示：

Table 3. When the number of neurons in the first layer is 8, the absolute value error ($\left|F(\vec{x}) - y_{w,b}(\vec{x})\right|$) of the network approximation to the ternary function

表 3. 第 1 层隐含层神经元数目为 8 个时，网络对该三元函数逼近的绝对值误差($\left|F(\vec{x}) - y_{w,b}(\vec{x})\right|$)

第 1 层隐含层节点数目/个 绝对值误差 $\left\|F(\vec{x}) - y_{w,b}(\vec{x})\right\|$ 第 2 层隐含层节点数目/个	8	9	10	11	12
8	0.00011	2.35e−05	1.80e−05	1.13e−05	1.86e−05
9	4.33e−05	1.76e−05	1.86e−05	2.38e−05	2.45e−05
10	3.82e−05	2.33e−05	3.96e−05	2.65e−05	1.22e−05
11	3.74e−05	3.66e−05	2.56e−05	2.05e−05	3.06e−05
12	6.67e−05	1.74e−05	7.26e−05	3.89e−05	3.70e−05

Table 4. When the number of neurons in the first layer is 9, the absolute value error ($\left|F\left(\vec{x}\right)-y_{w,b}\left(\vec{x}\right)\right|$) of the network approximation to the ternary function

表 4. 第 1 层隐含层神经元数目为 9 个时，网络对该三元函数逼近的绝对值误差($\left|F\left(\vec{x}\right)-y_{w,b}\left(\vec{x}\right)\right|$)

第 1 层隐含层节点数目/个 绝对值误差 $\left\|F\left(\vec{x}\right)-y_{w,b}\left(\vec{x}\right)\right\|$ 第 2 层隐含层节点数目/个	8	9	10	11	12
8	3.09e−05	1.94e−05	1.03e−05	3.42e−05	2.44e−05
9	1.55e−05	4.19e−05	2.37e−05	9.67e−05	9.99e−06
10	1.98e−05	1.72e−05	2.26e−05	0.000014	1.45e−05
11	0.00001	3.91e−05	2.45e−05	3.16e−05	2.08e−05
12	2.42e−05	1.90e−05	3.17e−05	1.47e−05	5.82e−05

Table 5. When the number of neurons in the first layer is 10, the absolute value error ($\left|F\left(\vec{x}\right)-y_{w,b}\left(\vec{x}\right)\right|$) of the network approximation to the ternary function

表 5. 第 1 层隐含层神经元数目为 10 个时，网络对该三元函数逼近的绝对值误差($\left|F\left(\vec{x}\right)-y_{w,b}\left(\vec{x}\right)\right|$)

第 1 层隐含层节点数目/个 绝对值误差 $\left\|F\left(\vec{x}\right)-y_{w,b}\left(\vec{x}\right)\right\|$ 第 2 层隐含层节点数目/个	8	9	10	11	12
8	1.49e−05	1.51e−05	1.51e−05	3.17e−05	2.60e−05
9	1.30e−05	0.000021	2.22e−05	1.46e−05	4.86e−05
10	1.87e−05	3.00e−05	9.98e−06	1,22e−05	1.84e−05
11	2.04e−05	3.15e−05	1.07e−05	1.06e−05	3.54e−05
12	5.74e−05	5.63e−05	1.29e−05	3.12e−05	1.8e−05

分析以表 3、表 4 和表 5 的数据，结合不同隐含层节点数目的网络逼近函数图观察得，三层隐含层神经元数目均为 10 时，网络的精度最高。

加入以上最适隐含层神经元数目后，创建训练该网络，然后输出训练后网络的权值矩阵和阈值矩阵如下：

$$iw1=\begin{bmatrix}0.0836 & -0.0225 & 1.4970\\ 0.3326 & -0.3347 & 0.0401\\ 0.0282 & -0.0558 & -0.5615\\ 0.0282 & -0.2551 & 0.0003\\ 0.0282 & -0.4149 & 0.0363\\ -0.3693 & 0.3385 & -0.0704\\ -0.3166 & -0.1202 & 0.0061\\ -0.0017 & -0.0395 & -0.5043\\ 1.0237 & 0.9184 & 0.8995\\ 0.1001 & 0.5558 & -0.0516\end{bmatrix},\quad b1=\begin{bmatrix}5.7915\\ -2.3379\\ -1.9746\\ 2.9009\\ 0.3804\\ 0.2849\\ -1.3349\\ 2.4243\\ 5.6608\\ -2.5790\end{bmatrix},\quad b2=\begin{bmatrix}0.1991\\ 5.7050\\ 1.0819\\ 3.5540\\ 0.4308\\ 3.9779\\ 3.7134\\ 5.0181\\ 0.0286\\ -0.2320\end{bmatrix},\quad b3=\begin{bmatrix}-2.4620\\ 5.0485\\ -4.7870\\ 3.4847\\ 3.4674\\ -0.5793\\ 2.5626\\ -2.5435\\ 2.6451\\ 1.1902\end{bmatrix}$$

$$lw2=\begin{bmatrix} 0.1489 & -1.2592 & -0.6252 & 0.6565 & -1.9817 & 2.3197 & 0.6367 & -1.4759 & 1.0787 & -2.2479 \\ -0.5982 & 4.6431 & 2.8179 & -1.9800 & -2.4183 & -0.9405 & 6.5865 & -3.4205 & -0.7129 & 2.1821 \\ 3.0404 & 0.3904 & 1.1372 & -0.5190 & -0.5646 & -2.8891 & 2.3506 & 0.0118 & -0.7921 & -0.4121 \\ 0.1027 & 0.8304 & -1.4192 & -1.6764 & -1.1641 & -0.5833 & -2.4755 & -2.2549 & 2.5575 & -0.9718 \\ 1.6900 & 3.1445 & 0.6508 & 3.0838 & -2.2925 & 0.4650 & 0.4518 & -4.4013 & -0.7923 & 0.0261 \\ 1.1173 & -1.3508 & -1.7383 & -2.5259 & -0.2260 & -0.7103 & -2.7721 & 3.4172 & -3.6923 & -1.5190 \\ 0.5790 & -3.4405 & 2.6726 & 0.4454 & 1.7916 & -0.6335 & -4.6042 & -3.6064 & 0.0606 & -1.8064 \\ 0.4809 & -1.0025 & -3.2142 & 0.8175 & -0.5893 & 0.2096 & 2.6041 & -3.4490 & -1.3181 & -2.4099 \\ -1.6148 & 0.4207 & 0.7285 & -2.5659 & -2.1103 & -2.4858 & 1.2063 & -0.9127 & 3.5801 & 1.9192 \\ 1.6121 & -2.7586 & 0.0855 & 1.7616 & 4.2070 & 1.3123 & 0.6139 & -2.6655 & -1.9198 & 0.3693 \end{bmatrix}$$

$$b4=2.5851$$

由上面计算得到的数值，可得如下神经网络函数：

$$y_{w,b}(\vec{x})=pureline\left(logsig\left(logsig\left(iw2*logsig\left(iw1*\vec{x}+b1\right)+b2\right)+b3\right)+b4\right) \tag{10}$$

从图 4 中(右)图可以看出逼近该三元函数的前馈神经网络的误差绝对值在$\left(0,10\times10^{-5}\right)$之间，结合其中原函数图像和网络输出结果对比分析可知，该网络逼近效果较好。

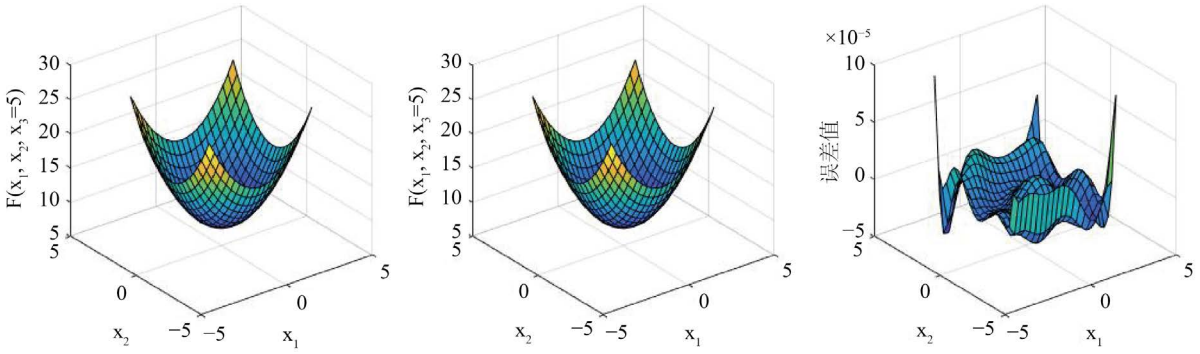

Figure 4. (left) original function image, (middle) output result diagram of the network, (right) absolute value error ($\left|F(\vec{x})-y_{w,b}(\vec{x})\right|$) diagram of the network approximation

图 4. (左) 原函数图像，(中) 该网络输出结果图，(右) 该网络逼近的绝对值误差($\left|F(\vec{x})-y_{w,b}(\vec{x})\right|$)图

4. 结语

本文通过数值计算结果证实前馈神经网络可用来逼近一元函数、二元函数、三元函数，能够达到较高的计算精度。发现输入样本的数量多少会影响神经网络计算的速度和精度，神经网络隐含层数目也会影响神经网络的逼近精度，建议同时改变输入样本的数量和隐含层数目来增加神经网络的逼近精度。本文的讨论具有一般性，适用于其他类人工神经网络在四元或四元以上的多元函数逼近问题的研究，也有助于理解相关人工神经网络的基本性质与作用。

参考文献

[1] 沈燮昌. 逼近论发展史简述(一) [J]. 数学研究及应用, 1982, 2(2): 171-180.

[2] 徐学良. 人工神经网络的发展及现状[J]. 微电子学, 2017, 47(2): 239-242.

[3] Bulsari, A. (1993) Some Analytical Solutions to the General Approximation Problem for Feedforward Neural Net-

works. *Neural Networks*, **6**, 991-996. https://doi.org/10.1016/S0893-6080(09)80008-7

[4] Suzuki, S. (1998) Constructive Function-Approximation by Three-Layer Artificial Neural Networks. *Neural Networks*, **11**, 1049-1058. https://doi.org/10.1016/S0893-6080(98)00068-9

[5] Twomey, J.M. and Smith, A.E. (1998) Bias and Variance of Validation Methods for Function Approximation Neural Networks under Conditions of Sparse Data. *IEEE Transactions on Systems, Man and Cybernetics, Part C* (*Applications and Reviews*), **28**, 417-430. https://doi.org/10.1109/5326.704579

[6] Ferrari, S. and Stengel, R.F. (2005) Smooth Function Approximation Using Neural Networks. *IEEE Transactions on Neural Networks*, **16**, 24-38. https://doi.org/10.1109/TNN.2004.836233

[7] Yao, S., Wei, C.J. and He, Z.Y. (2013) Evolving Wavelet Neural Networks for Function Approximation. *Electronics Letters*, **17**, 586-594.

[8] 韦岗, 李华, 徐秉铮. 关于前馈多层神经网络多维函数逼近能力的一个定理[J]. 电子与信息学报, 1997, 19(4): 433-438.

[9] 王强, 余岳峰, 张浩炯. 利用人工神经网络实现函数逼近[J]. 计算机仿真, 2002(5): 44-47.

[10] 侯木舟. 基于构造型前馈神经网络的函数逼近与应用[D]: [博士学位论文]. 长沙: 中南大学, 2009.

[11] 李鹏柱. 关于神经网络与样条函数的逼近性能研究[D]: [硕士学位论文]. 银川: 宁夏大学, 2016.

[12] 许洋. 前馈型神经网络算法优化分析[J]. 硅谷, 2014(13): 66, 42.

[13] Heaton, J.B., Polson, N.G. and Witte, J.H. (2018) Deep Learning in Finance.

[14] 周开利. 神经网络模型及其 MATLAB 仿真程序设计[M]. 北京: 清华大学出版社, 2005.

[15] 孙永生. 函数逼近论[M]. 北京: 北京师范大学出版社, 1989.

[16] 李晓东, 胡志恒, 虞厥邦. 一种前馈神经网络的快速学习算法[J]. 信号处理, 2004, 20(2): 184-187.

[17] Foresee, F.D. and Hagan, M.T. (1997) Gauss-Newton Approximation to Bayesian Regularization. *International-Joint Conference on Neural Network*, 1930-1935.