多元多项式函数的三层前向神经网络逼近方法

王建军[1),2)] 徐宗本[2)]

[1)](西南大学数学与统计学院 重庆 400715)

[2)](西安交通大学信息与系统科学研究所 西安 710049)

摘 要 该文首先用构造性方法证明:对任意 r 阶多元多项式,存在确定权值和确定隐元个数的三层前向神经网络,它能以任意精度逼近该多项式,其中权值由所给多元多项式的系数和激活函数确定,而隐元个数由 r 与输入变量维数确定.作者给出算法和算例,说明基于文中所构造的神经网络可非常高效地逼近多元多项式函数.具体化到一元多项式的情形,文中结果比曹飞龙等所提出的网络和算法更为简单、高效;所获结果对前向神经网络逼近多元多项式函数类的网络构造以及逼近等具有重要的理论与应用意义,为神经网络逼近任意函数的网络构造的理论与方法提供了一条途径.

关键词 前向神经网络;多元多项式;逼近;算法

中图法分类号 TP18 **DOI 号**: 10.3724/SP.J.1016.2009.02482

Approximation Method of Multivariate Polynomials by Feedforward Neural Networks

WANG Jian-Jun[1),2)] XU Zong-Ben[2)]

[1)](*School of Mathematics & Statistics, Southwest University, Chongqing* 400715)

[2)](*Institute for Information and System Science, Xi'an Jiaotong University, Xi'an* 710049)

Abstract Firstly, this paper investigates that for a given multivariate polynomials with r order, a three-layer feedforward neural networks with determinate weights and the number of hidden-layer nodes can be established by a constructive method to approximate the polynomials to any degree of accuracy. Secondly, the weights are decided by both the coefficients of the polynomials and the activation function, and the number of hidden-layer nodes of the constructed network depends on the order of approximating polynomial and the dimension of input on the network. Then the algorithm and algorithmic examples are given, where the constructed networks can very efficiently approximate multivariate polynomials. Specifically, for a univariate polynomial, the constructed network and realization of algorithm obtained are simpler and more efficient than the methods proposed by Cao Fei-Long in 2003. The obtained results are of theoretical and practical importance in constructing a feedforward neural network with three-layer to approximate the class of multivariate polynomials. They also provide a route in both theory and method of constructing neural network to approximate any multivariate functions.

Keywords feedforward neural network; multivariate polynomials; approximation; algorithm

---

收稿日期:2006-07-05;最终修改稿收到日期:2009-03-30.本课题得到国家"九七三"重点基础研究发展规划项目基金(2007CB311000)、国家自然科学基金重点项目(70531030)、国家自然科学基金(10726040,10701062,10826081)、教育部科学技术重点项目(108176)、重庆市科委自然科学基金计划资助项目(CSTC,2009BB2306)、中国博士后科学基金(20080431237)、西南大学博士基金(SWUB2007006)和西南大学发展基金(SWUF2007014)资助.**王建军**,男,1976 年生,博士,副教授,主要研究方向为神经网络、学习理论和逼近论.E-mail:wjj@swu.edu.cn.**徐宗本**,男,1955 年生,博士,教授,博士生导师,主要研究领域为人工智能、非线性泛函分析等.

1 引　言

近年来,许多学者对神经网络逼近问题进行了研究,取得了一系列重要成果.神经网络已经在工程、计算机、物理、生物等学科中得到了广泛的应用,大多数应用都被转化为利用神经网络逼近多元函数的问题(如文献[1-5]等).神经网络之所以能得到广泛应用,其主要原因之一是它具有一定意义上的万有逼近性(如文献[6-7]等).所有这些研究的一个典型结论是:任何一个定义在 R^d 上的连续函数可以通过一个具有单一隐层的三层前向神经网络任意逼近.一个具有单一隐层,含 d 个输入、1 个输出的三层前向神经网络数学上可表示为

$$N(\boldsymbol{x})=\sum_{i=1}^{m}c_i\sigma\Big(\sum_{j=1}^{d}w_{ij}x_j+\theta_i\Big),\ \boldsymbol{x}\in R^d,d\geq 1 \quad (1)$$

其中 $1\leq i\leq m,\theta_i\in R$ 是阈值,$w_i=(w_{i1},w_{i2},\cdots,w_{id})^{\mathrm{T}}\in R^d$ 是输入层与隐层第 i 个神经元的连接权值,c_i 是隐层与输出层之间的连接权值,σ 是隐层节点的激活函数(传递函数).通常情况下,网络激活函数 σ 取为 sigmoid 型函数,即满足 $\sigma(t)\to 1(t\to\infty)$,$\sigma(t)\to 0(t\to-\infty)$的函数.用向量形式表示,式(1)可进一步表达为

$$N(\boldsymbol{x})=\sum_{i=1}^{n}c_i\sigma(\boldsymbol{w}_i\cdot\boldsymbol{x}+\theta_i),\ \boldsymbol{x}\in R^d.$$

众所周知,人工神经网络的结构设计(使之有能力学习给定的函数)是其应用中的重要而基本的问题.最近,有较多的工作(如文献[8-10])研究前向神经网络的逼近精度与隐元个数的关系,以从理论上反映网络逼近速度与网络拓扑之间的关系.但是,这些理论结果并没有给出实现函数逼近的具体算法,所构造的网络也过于复杂,不易实现,所以很难在实际中得到应用.一元多项式是最简单和最基本的被逼近函数形式.在文献[11]中,作者对一元多项式构造了一种前向神经网络,给出了逼近的理论结果和算法实现;对于多元情况,由于多元区域中点的方向的无穷性、多项式的展开分解以及差分的介入等问题的复杂性,它并不能表示为一元多项式的简单叠加,在逼近意义下,也不是一元多项式的简单推广,因而对于多元多项式,神经网络实现起来并不容易.然而,我们知道,多元多项式能够任意逼近任何一个连续多元函数,因而如何高效实现多元多项式的神经网络逼近对于发展对一般函数的神经网络逼近(特别是网络设计理论)有重要意义.有鉴于此,本文研究目标函数为多元多项式的三层前向神经网络的逼近问题.我们将给出一个具有确定隐元个数和确定权值向量的三层前向神经网络来实现对多元多项式的任意逼近.所给出的确定网络,其隐层节点数由所逼近多项式的阶数 r 和输入空间的维数 d 确定,而权值由所逼近多项式的系数和激活函数确定.我们将给出一个具体的算法实现网络设计.算例表明:所提出的算法十分高效,在一元情况,同文献[11]的结果相比,本文算法所构造网络的逼近精度比文献[11]提高了 10 倍.本文所获结果对前向神经网络逼近多元多项式函数类的网络具体构造以及实现逼近的方法等问题具有重要的指导意义.

2 记号及主要结果

在本文中我们将采用如下记号 Z_0,R 分别表示非负整数、实数,R^d 表示 d 维实 Euclid 空间;Z_0^d 表示 $\overbrace{Z_0\times Z_0\times\cdots\times Z_0}^{d\text{个}}$. 对任何 $\boldsymbol{x}=(x_1,x_2,\cdots,x_d)$,$\boldsymbol{y}=(y_1,y_2,\cdots,y_d)\in R^d$,$\boldsymbol{j}=(j_1,j_2,\cdots,j_d)$,$\boldsymbol{q}=(q_1,q_2,\cdots,q_d)\in Z_0^d$. 向量 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 的内积表示为

$$\boldsymbol{x}\cdot\boldsymbol{y}=\sum_{k=1}^{d}x_k y_k,$$

同时记

$$\boldsymbol{x}^{\boldsymbol{j}}=\prod_{i=1}^{d}x_i^{j_i}.$$

用 $\boldsymbol{j}\leq\boldsymbol{q}$ 表示 $j_i\leq q_i(i=1,2,\cdots,d)$,$|\boldsymbol{j}|\leq|\boldsymbol{q}|$ 表示 $\sum_{i=1}^{d}j_i\leq\sum_{i=1}^{d}q_i$. 对任意定义在 R^d 上的光滑函数 f,其 $|\boldsymbol{j}|$-阶偏导数表示如下:

$$f^{(\boldsymbol{r})}(\boldsymbol{x}):=\frac{\partial^{|\boldsymbol{r}|}f}{\partial\boldsymbol{x}^{|\boldsymbol{r}|}}(\boldsymbol{x})=\frac{\partial^{|\boldsymbol{r}|}f}{\partial x_1^{r_1}\partial x_2^{r_2}\cdots\partial x_d^{r_d}}(\boldsymbol{x}),$$

其中 $|\boldsymbol{r}|=r_1+r_2+\cdots+r_d$.

我们用 $P_r(d)$ 表示定义在有界区域 S 上的所有 d 元实的、次数不超过 $|\boldsymbol{r}|$ 的代数多项式.我们利用这些记号给出如下的逼近定理.

定理. 设 ϕ 是定义在 R 上的具有 $|\boldsymbol{r}|+1$-阶连续有界导数的函数,且对任意的 $k\in Z_0$,$0\leq k\leq|\boldsymbol{r}|+1$,存在某一 $\theta\in R$,使得 $\phi^{(k)}(\theta)\neq 0$. $p_r(\boldsymbol{x})\in P_r(d)$,则可以构造一个输入、一个输出及隐元个数为 $n=\sum_{0\leq|\boldsymbol{j}|\leq|\boldsymbol{r}|}\prod_{i=1}^{d}(j_i+1)$ 的三层前向神经网络:

$N_n(\boldsymbol{x})=\sum_{0\leq|\boldsymbol{j}|\leq|\boldsymbol{r}|}\sum_{0\leq\boldsymbol{i}\leq\boldsymbol{j}}c_{\boldsymbol{i},\boldsymbol{j}}\phi(\boldsymbol{w}\cdot\boldsymbol{x}+\theta)$,$c_{\boldsymbol{i},\boldsymbol{j}}\in R$,$\boldsymbol{w}\in R^d$,使得

$$|N_n(\boldsymbol{x})-P_r(\boldsymbol{x})|<\varepsilon,\ \forall\varepsilon>0.$$

证明. 设

$$p_r(\boldsymbol{x})=\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}a_j\boldsymbol{x}^j \tag{2}$$

记 $M_i=\{|x_i|_{\max},\ \boldsymbol{x}=(x_1,x_2,\cdots,x_d)\in S\}$, $i=1,2,\cdots,d$. 由于[9]

$$\begin{aligned}\phi^{(j)}(\boldsymbol{\omega}\cdot\boldsymbol{x}+\theta)&=\frac{\partial^{|j|}}{\partial\omega_1^{j_1}\partial\omega_2^{j_2}\cdots\partial\omega_d^{j_d}}[\phi(\boldsymbol{\omega}\cdot\boldsymbol{x}+\theta)]\\&=\boldsymbol{x}^j\phi^{(|j|)}(\boldsymbol{\omega}\cdot\boldsymbol{x}+\theta)\end{aligned} \tag{3}$$

于是

$$\phi_{j,x}(\theta)=\phi^{(j)}(\boldsymbol{\omega}\cdot\boldsymbol{x}+\theta)|_{\omega=0}=\boldsymbol{x}^j\phi^{(|j|)}(\theta) \tag{4}$$

因而

$$\boldsymbol{x}^j=\frac{\phi_{j,x}(\theta)}{\phi^{(|j|)}(\theta)} \tag{5}$$

将式(5)代入式(2),我们得到

$$p_r(\boldsymbol{x})=\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}a_j\frac{\phi_{j,x}(\theta)}{\phi^{(|j|)}(\theta)} \tag{6}$$

对任意固定的 $b\in R$,我们考虑以下有限 j-阶差分

$$\Delta_{h,x}^j\phi(\theta)=\sum_{0\le\boldsymbol{i}\le\boldsymbol{j}}(-1)^{|i|}\binom{\boldsymbol{j}}{\boldsymbol{i}}\phi(h(2\boldsymbol{i}-\boldsymbol{j})^{\mathrm{T}}\cdot\boldsymbol{x}+\theta) \tag{7}$$

其中 $\binom{\boldsymbol{j}}{\boldsymbol{i}}=\prod_{k=1}^{d}\binom{j_k}{i_k}$. 由三角不等式以及差分的积分表示,我们得到

$$\begin{aligned}&\left|p_r(\boldsymbol{x})-\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}a_j\frac{1}{\phi^{(|j|)}(\theta)}(2h)^{-|j|}\Delta_{h,x}^j\phi(\theta)\right|\\&=\left|\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}a_j\frac{\phi_{j,x}(\theta)}{\phi^{(|j|)}(\theta)}-\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}a_j\frac{1}{\phi^{(|j|)}(\theta)}(2h)^{-|j|}\Delta_{h,x}^j\phi(\theta)\right|\\&\le\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}|a_j|\frac{1}{|\phi^{(|j|)}(\theta)|}|\phi_{j,x}(\theta)-(2h)^{-|j|}\Delta_{h,x}^j\phi(\theta)|\\&=\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}|a_j|\frac{1}{|\phi^{(|j|)}(\theta)|}\Bigg|\boldsymbol{x}^j\phi^{(|j|)}(\theta)-\boldsymbol{x}^j(2h)^{-|j|}\int_{-h}^{h}\int_{-h}^{h}\cdots\int_{-h}^{h}\phi^{(|j|)}(\theta+(\tau_1+\tau_2+\cdots+\tau_{j_1})x_1+\cdots+(\tau_{|j|-j_d+1}+\cdots+\tau_{|j|})x_d)\mathrm{d}\tau\Bigg|\\&=\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}|a_j|\frac{1}{|\phi^{(|j|)}(\theta)|}\boldsymbol{x}^j\Bigg|(2h)^{-|j|}\cdot\int_{-h}^{h}\int_{-h}^{h}\cdots\int_{-h}^{h}\phi^{(|j|)}(\theta)\mathrm{d}\tau-(2h)^{-|j|}\int_{-h}^{h}\int_{-h}^{h}\cdots\int_{-h}^{h}\phi^{(|j|)}(\theta+(\tau_1+\tau_2+\cdots+\tau_{j_1})x_1+\cdots+(\tau_{|j|-j_d+1}+\cdots+\tau_{|j|})x_d)\mathrm{d}\tau\Bigg|\\&\le\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}|a_j|\frac{1}{|\phi^{(|j|)}(\theta)|}\boldsymbol{x}^j\Omega(\phi^{(|j|)},M_1|\boldsymbol{j}|(2h))\\&\le2M_1h\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}|a_j|\prod_{i=1}^{d}M_i^{j_i}\frac{\|\phi^{(|j|+1)}\|_\infty}{|\phi^{(|j|)}(\theta)|}|\boldsymbol{j}|\end{aligned} \tag{8}$$

其中 $\Omega(\phi,\delta)=\sup_{|t-x|<\delta}|\phi(\boldsymbol{x})-\phi(t)|$(见文献[12])是函数的连续模,且当 ϕ 有连续导数时,$\Omega(\phi,\delta)\le\delta\|\phi\|_\infty$($\|\phi\|_\infty=\sup_x|\phi(\boldsymbol{x})|$). 令 $M_0=\max\{|\phi^{(i)}(\theta)|,i=0,1,\cdots,|\boldsymbol{r}|+1\}$,于是

$$\begin{aligned}&\left|p_r(\boldsymbol{x})-\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}a_j\frac{1}{\phi^{(|j|)}(\theta)}(2h)^{-|j|}\Delta_{h,x}^j\phi(\theta)\right|\\&\le2M_1M_0h\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}\frac{|a_j||\boldsymbol{j}|}{|\phi^{(|j|)}(\theta)|}\prod_{i=1}^{d}M_i^{j_i}\le Mh\end{aligned} \tag{9}$$

其中常数

$$M=2M_1M_0\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}\frac{|a_j||\boldsymbol{j}|}{|\phi^{(|j|)}(\theta)|}\prod_{i=1}^{d}M_i^{j_i}.$$

由式(8)、(9),我们可以构造出如下神经网络

$$N_n(\boldsymbol{x})=\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}\sum_{0\le\boldsymbol{i}\le\boldsymbol{j}}c_{i,j}\phi(\boldsymbol{w}^{\mathrm{T}}\cdot\boldsymbol{x}+\theta) \tag{10}$$

其中

$$\boldsymbol{w}=h(2\boldsymbol{i}-\boldsymbol{j}) \tag{11}$$

$$c_{i,j}=a_j\frac{1}{\phi^{(|j|)}(\theta)}(2h)^{-|j|}(-1)^{|i|}\binom{\boldsymbol{j}}{\boldsymbol{i}} \tag{12}$$

且由式(9)知 $N_n(\boldsymbol{x})$ 满足

$$|p_r(\boldsymbol{x})-N_n(\boldsymbol{x})|\le Mh.$$

令 $h<\frac{\varepsilon}{M}$,得到

$$|p_r(\boldsymbol{x})-N_n(\boldsymbol{x})|<\varepsilon.$$

至于隐层单元个数,我们由式(10)很容易看出,网络具有 $\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}\prod_{i=1}^{d}(j_i+1)$ 个单元.

注 1. 从上述证明中我们看到,对于给定的多元多项式 $p_r(\boldsymbol{x})$,其神经网络的权值可具体由式(11)和(12)确定. 从式(11)和(12)我们看到,它由所逼近多项式的系数和所选定的网络激活函数在 θ 点的各阶导数值唯一确定.

3 算法和算例

总结上节讨论,我们能给出如下构造逼近多元多项式神经网络的算法.

算法 1.

给定参数:多元多项式的系数 a_j,阶数 r;

误差要求 ε,输入最大值 $M_i=x_{i\max}$, $i=1,2,\cdots,d$;满足要求的激活函数 ϕ 的具体表达式;搜索步长 Δ.

1. 求出隐元个数 $\sum_{0\le|\boldsymbol{j}|\le|\boldsymbol{r}|}\prod_{i=1}^{d}(j_i+1)$;

2. 选择阈值 θ 并计算 $\phi^{(|j|)}(\theta)$;

3. 求出

$$M_0=\max\{|\phi^{(j)}(\theta)|,j=0,1,\cdots,|\boldsymbol{r}|+1\};$$

4. 计算

$$M=2M_1M_0\sum_{0\leq|\boldsymbol{j}|\leq|\boldsymbol{r}|}\frac{|a_j|\,|\boldsymbol{j}|}{|\phi^{(|\boldsymbol{j}|)}(\theta)|}\prod_{i=1}^{d}M_i^{j_i};$$

5. 选取 h 满足 $h<\frac{\varepsilon}{M}$ 并令

$$\boldsymbol{w}=h(2\boldsymbol{i}-\boldsymbol{j});$$

6. 利用方程(12)计算权 $c_{i,j}$，即

$$c_{i,j}=a_j\frac{1}{\phi^{(|\boldsymbol{j}|)}(\theta)}(2h)^{-|\boldsymbol{j}|}(-1)^{|\boldsymbol{i}|}\binom{\boldsymbol{j}}{\boldsymbol{i}};$$

7. 结束.

下面给出算例.

首先,我们采用文献[11]中的例子作为我们第一个算例,且和文献[11]的结果进行比较.

例 1.　选取激活函数 $\varphi(x)=\frac{1}{1+e^{-x}}$，令被逼近的多项式函数为 $P_1(x)=1-3x$，输入的最大值为 $M_1=x_{\max}=10$,误差要求 $\varepsilon=0.001$,则节点数是3,选取阈值 θ 使得 $e^{-\theta}=3$,通过计算,得到 $\varphi(\theta)=\frac{1}{4}$，$\varphi^{(1)}(\theta)=\frac{3}{16}$，$M_0<1$,所以,

$$M\leq 3.2\times10^3.$$

于是,我们可以选择 $h=10^{-7}<\frac{\varepsilon}{M}$,注意到 $c_{i,j}$ 的表达式，我们计算得到

$$c_{0,0}=4,\ c_{0,1}=-8\times10^7,\ c_{1,1}=8\times10^7.$$

从而,对于多项式函数 $P_1(x)=1-3x$，我们可以构造前向神经网络

$$N(x)=c_{0,0}\phi(\theta)-8\times10^7\phi(-10^{-7}x+\theta)+8\times10^7\phi(10^{-7}x+\theta),$$

计算结果见表 1,误差曲线见图 1,文献[11]中关于此例的误差曲线见图 2.

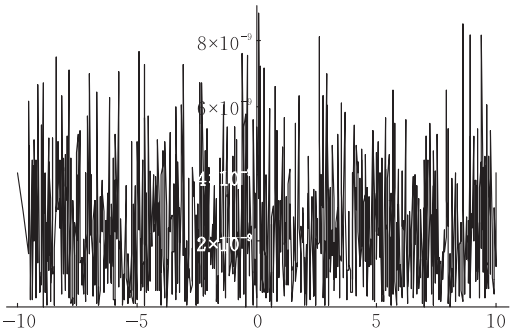

图 1　例 1 误差曲线图

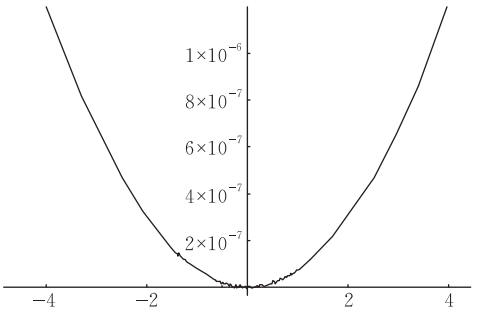

图 2　文献[11]的误差曲线图

表　1

No.	x	$P_1(x)$	$N(x)$	$\lvert N(x)-P_1(x)\rvert$	$N'(x)$	$\lvert N'(x)-P_1(x)\rvert$
1	0.0	1.0000000000	1.0000000000	0.0000000000	1.0000000000	0.0000000000
2	0.1	0.7000000000	0.70000000404369	0.00000000404369	0.7000000029	0.0000000029
3	0.2	0.4000000000	0.39999999920559	0.00000000079441	0.3999999910	0.0000000090
4	0.3	0.1000000000	0.10000000324928	0.00000000324928	0.0999999940	0.0000000060
5	0.4	−0.2000000000	−0.20000000158882	0.00000000158882	−0.2000000104	0.0000000104
6	0.5	−0.5000000000	−0.49999999754513	0.00000000245487	−0.5000000149	0.0000000149
7	0.6	−0.8000000000	−0.80000000238323	0.00000000238323	−0.8000000268	0.0000000268
8	0.7	−1.1000000000	−1.09999999833954	0.00000000166046	−1.1000000312	0.0000000312
9	0.8	−1.4000000000	−1.40000000317764	0.00000000317764	−1.4000000506	0.0000000506
10	0.9	−1.7000000000	−1.70000000357484	0.00000000357484	−1.7000000625	0.0000000625
11	1.0	−2.0000000000	−1.99999999731071	0.00000000268929	−1.7000000670	0.0000000670
12	1.1	−2.3000000000	−2.30000000436925	0.00000000436925	−1.7000000938	0.0000000938
13	1.2	−2.6000000000	−2.59999999810512	0.00000000189488	−2.6000001057	0.0000001057
14	1.3	−2.9000000000	−2.89999999850232	0.00000000149768	−2.9000001251	0.0000001251
15	1.4	−3.2000000000	−3.19999999889953	0.00000000110047	−3.2000001445	0.0000001445
16	1.5	−3.5000000000	−3.49999999929673	0.00000000070327	−3.5000001713	0.0000001713
17	1.6	−3.8000000000	−3.79999999747349	0.00000000252651	−3.8000001907	0.0000001907
18	1.7	−4.1000000000	−4.10000000453203	0.00000000453203	−4.1000002175	0.0000002175
19	1.8	−4.4000000000	−4.39999999826790	0.00000000173210	−4.4000002443	0.0000002443
20	1.9	−4.7000000000	−4.69999999866511	0.00000000133489	−4.7000002673	0.0000002673
21	2.0	−5.0000000000	−5.00000000128276	0.00000000128276	−5.0000002980	0.0000002980
⋮	⋮	⋮	⋮	⋮	⋮	⋮
95	9.4	−27.2000000000	−27.19999999514877	0.00000000485123	−27.2000066265	0.0000066265
96	9.5	−27.5000000000	−27.49999999776642	0.00000000223358	−27.5000067725	0.0000067725
97	9.6	−27.8000000000	−27.80000000038407	0.00000000038407	−27.8000069111	0.0000069111
98	9.7	−28.1000000000	−28.10000000300173	0.00000000300172	−28.1000070646	0.0000070646

（续 表）

No.	x	$P_1(x)$	$N(x)$	$\lvert N(x)-P_1(x)\rvert$	$N'(x)$	$\lvert N'(x)-P_1(x)\rvert$
99	9.8	−28.4000000000	−28.39999999895804	0.00000000104196	−28.4000072032	0.0000072032
100	9.9	−28.7000000000	−28.69999999713480	0.00000000286521	−28.7000073492	0.0000073492
⋮	⋮	⋮	⋮	⋮	⋮	⋮
193	−9.2	28.6000000000	28.60000000101570	0.00000000101570	28.5999936535	0.0000063465
194	−9.3	28.9000000000	28.90000000363335	0.00000000363335	28.8999935165	0.0000064835
195	−9.4	29.2000000000	29.19999999514877	0.00000000485123	29.1999933645	0.0000066355
196	−9.5	29.5000000000	29.49999999776642	0.00000000223358	29.4999932199	0.0000067801
197	−9.6	29.8000000000	29.80000000038407	0.00000000038408	29.7999930903	0.0000069097
198	−9.7	30.1000000000	30.10000000300173	0.00000000300173	30.0999929457	0.0000070543
199	−9.8	30.4000000000	30.39999999895804	0.00000000104196	30.3999927937	0.0000072063
200	−9.9	30.7000000000	30.69999999713480	0.00000000286521	30.6999926418	0.0000073582
201	−10	31.0000000000	30.99999999753200	0.00000000246800	30.9999924972	0.0000075038

注：表 1 中 $N(x)$是例 1 用本文方法所构造的网络，$N'(x)$是文献[11]所构造的网络.

注 2. 从这个例子可以看出，我们所构造的网络不但简单、容易计算网络权值，而且逼近精度非常理想.从网络构造上来说，本文所构造的网络比文献[11]的网络更容易实现，其计算复杂度明显下降，这只需注意到，文献[11]中需要通过计算以下矩阵的逆

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & r \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 1^r & \cdots & r^r \end{pmatrix}^{-1}$$

来实现网络构造，而显然，当多项式的阶数很高时，上式计算复杂度甚高；而本文的算法不涉及这样的矩阵求逆运算. 从计算结果来看，本文的结果明显好于文献[11]的结果，其误差精度提高了 10 倍.

例 2. 选取激活函数 $\varphi(x)=\dfrac{1}{1+\mathrm{e}^{-x}}$，令被逼近函数为二元三次多项式 $P_2(x)=6x_1-3x_1x_2+x_2^3$，输入的最大值为 $M_1=M_2=10$，误差要求 $\varepsilon=0.001$，则节点数 $n=30$，并且 $\varphi(\theta)=\dfrac{1}{4}$，$\varphi^{(1)}(\theta)=\dfrac{3}{16}$，$\varphi^{(2)}(\theta)=\dfrac{3}{32}$，$\varphi^{(3)}(\theta)=-\dfrac{3}{128}$，$\varphi^{(4)}(\theta)=-\dfrac{15}{128}$，以 $M_0=\dfrac{1}{4}$，从而 $M=6.736\times10^5$. 我们取 $h=10^{-9}<\dfrac{\varepsilon}{M}$，则利用式(10)，我们可以得到

$$\begin{aligned} N(x)=&16h^{-1}(\varphi(hx_1+\theta)-\varphi(-hx_1+\theta))-\\ &8h^{-2}(\varphi(h(x_2+x_1)+\theta)-\varphi(-h(x_1-x_2)+\theta)-\\ &\varphi(-h(x_2-x_1)+\theta)+\varphi(-h(x_2+x_1)+\theta))-\\ &16/3h^{-3}(\varphi(3hx_2+\theta)-3\varphi(hx_2+\theta)+\\ &3\varphi(-hx_2+\theta)-\varphi(-3hx_2+\theta)) \end{aligned} \tag{13}$$

使得

$$\lvert P_2(x)-N(x)\rvert\leq0.001.$$

表 2(以搜索步长 $\Delta=1.0$)和图 3(误差曲线图($h=10^{-4}$))对例 2 进一步加以说明.

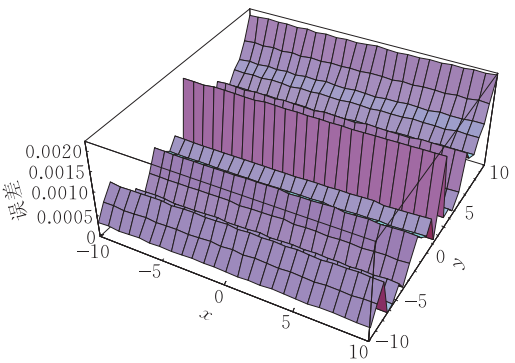

图 3 例 2 的误差曲面图($h=10^{-4}$)

表 2

No.	x_1	x_2	$N(x)$	$P_2(x)$	$\lvert N(x)-P_2(x)\rvert$
1	0.0	0.0	0	0	0
2	0.0	1.0	1.0000000000000000016	1.0000000000000000000	1.6×10^{-18}
3	0.0	2.0	8.0000000000000000520	8.0000000000000000000	5.20×10^{-17}
4	0.0	3.0	27.000000000000000395	27.000000000000000000	3.95×10^{-16}
5	0.0	4.0	64.000000000000001664	64.000000000000000000	1.664×10^{-15}
6	0.0	5.0	125.00000000000000508	125.00000000000000000	5.08×10^{-15}
7	0.0	6.0	216.00000000000001264	216.00000000000000000	1.264×10^{-14}
8	0.0	7.0	343.00000000000002731	343.00000000000000000	2.731×10^{-14}
9	0.0	8.0	512.00000000000005325	512.00000000000000000	5.325×10^{-14}
10	0.0	9.0	729.00000000000009595	729.00000000000000000	9.595×10^{-14}
11	0.0	10.0	1000.0000000000001625	1000.0000000000000000	1.625×10^{-13}
12	1.0	0.0	5.9999999999999999999	6.0000000000000000000	1.0×10^{-19}
13	1.0	1.0	4.0000000000000000028	4.0000000000000000000	2.8×10^{-18}

（续　表）

No.	x_1	x_2	$N(x)$	$P_2(x)$	$\lvert N(x)-P_2(x)\rvert$
14	1.0	2.0	8.0000000000000000581	8.0000000000000000000	5.81×10^{-17}
15	1.0	3.0	24.000000000000000414	24.000000000000000000	4.14×10^{-16}
16	1.0	4.0	58.000000000000001706	58.000000000000000000	1.706×10^{-15}
17	1.0	5.0	116.00000000000000516	116.00000000000000000	5.16×10^{-15}
⋮	⋮	⋮	⋮	⋮	⋮
110	9.0	10.0	784.00000000000017259	784.00000000000000000	1.7259×10^{-13}
111	10.0	0.0	59.999999999999999875	60.000000000000000000	1.25×10^{-16}
112	10.0	1.0	31.000000000000000508	31.000000000000000000	5.08×10^{-16}
113	10.0	2.0	8.0000000000000012270	8.0000000000000000000	1.2270×10^{-15}
114	10.0	3.0	−2.9999999999999976864	−3.0000000000000000000	2.3136×10^{-15}
115	10.0	4.0	4.0000000000000044390	4.0000000000000000000	4.4390×10^{-15}
116	10.0	5.0	35.000000000000008859	35.000000000000000000	8.859×10^{-15}
⋮	⋮	⋮	⋮	⋮	⋮
343	−1.0	2.0	8.0000000000000000459	8.0000000000000000000	4.59×10^{-17}
344	−1.0	3.0	30.000000000000000376	30.000000000000000000	3.76×10^{-16}
345	−1.0	4.0	70.000000000000001622	70.000000000000000000	1.622×10^{-15}
346	−1.0	5.0	134.00000000000000500	134.00000000000000000	5×10^{-15}
⋮	⋮	⋮	⋮	⋮	⋮
437	−10.0	6.0	336.00000000000000766	336.00000000000000000	7.66×10^{-15}
438	−10.0	7.0	493.00000000000002092	493.00000000000000000	2.092×10^{-14}
439	−10.0	8.0	692.00000000000004517	692.00000000000000000	4.517×10^{-14}
440	−10.0	9.0	939.00000000000008590	939.00000000000000000	8.590×10^{-14}
441	−10.0	10.0	1240.0000000000001501	1240.0000000000000000	1.501×10^{-13}

注:表 2 中 $N(x)$是例 2 所构造的网络.

注 3. 从例 2 可以看出,我们构造的网络实际用到的隐元个数是 10(由式(13),$N(x)$右端共有 10 项是非零的,故此时隐元个数为 10),而不是像定理所预测的 $30(\sum_{j_1=0}^{1}\sum_{j_2=0}^{3}(j_1+1)\cdot(j_2+1)=30)$,这是由于我们的定理是对于一般的多元多项式来给出隐元个数,而本例仅是一般多项式的一个特例. 同时我们知道,对多元多项式,由于其方向的多样性,并不是简单的一元多项式的叠加,比一维情况复杂得多;从这个算例我们可以看到,我们所构造的神经网络简单,非常容易计算,实现了对多元多项式的逼近,误差结果十分理想,它为我们实现对任意多元函数的逼近提供了一个很好的范例.

例 3.　选取激活函数 $\varphi(x)=\dfrac{1}{1+e^{-x}}$,多项式 $P_3(x,y)=x^3-3x^2y+3xy^2-y^3$,用我们的方法得到的网络逼近 $P_3(x,y)$的仿真曲面图和误差曲面分别见图 4、图 5.

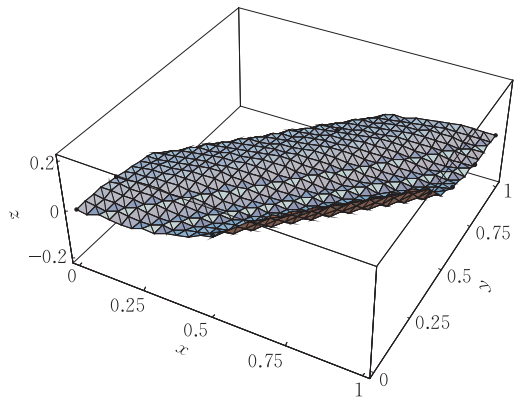

图 4　神经网络对 $P_3(x,y)$的仿真曲面图

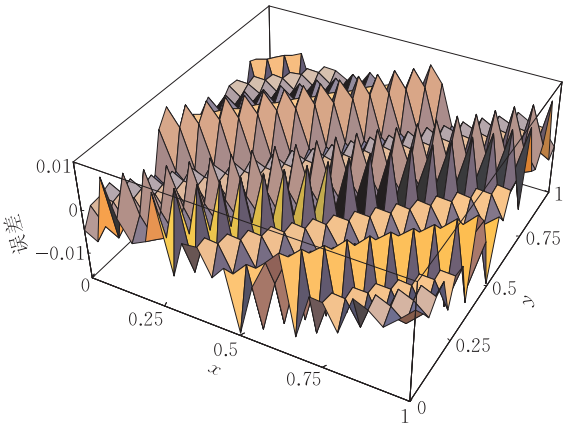

图 5　例 3 的误差曲面图($h=10^{-4.05}$)

4　结　论

本文所构造的三层前向人工神经网络的方法和逼近的具体算法实用简单,容易计算. 通过算例看出实现这一逼近比较容易,而且十分高效,计算复杂度较低. 所获结果表明:对给定阶数 r 的多元多项式,存在确定权值和确定隐元个数的三层前向神经网络,它能以任意精度逼近该多项式,其中权值由所给多元多项式的系数和激活函数确定,而隐元个数由 r 与输入变量维数确定,而且这一逼近完全可以

通过一个具体的算法实现,这为我们对任意函数被神经网络逼近提供了一个很好的理论和实践方法,具有重要的指导意义.

致 谢 作者衷心感谢审稿人提出的宝贵意见和建议!

参 考 文 献

[1] Mckenna T, Davis J, Zometzer S F. Single Neuron Computation. New York: Academic Press, 1992

[2] Skrzypek J, Karplus W. Neural Networks in Vision and Pattern Recognition. New Jersey: World Scientific, 1992

[3] Taylo J G. Mathematical Approaches to Neural Networks. New York: North-Holland, 1993

[4] Chen Tian-Ping, Chen Hong. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to a dynamic system. IEEE Transactions on Neural Networks, 1995, 6(4): 911-917

[5] Chen Tian-Ping. Approximation problems in system identification with neural networks. Science in China, Series A, 1994, 24(1): 1-7(in Chinese)
(陈天平. 神经网络及其在系统识别应用中的逼近问题. 中国科学(A 辑), 1994, 24(1): 1-7)

[6] Cybenko G. Approximation by superpositions of sigmoidal function. Mathematics of Control. Signals and Systems, 1989, 2(4): 303-314

[7] Hornik K, Stinchombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks, 1989, 2(2): 359-366

[8] Kurkova V, Kainen P C, Kreinovich V. Estimates of the number of hidden units and variation with respect to half-space. Neural Networks, 1997, 10(6): 1068-1078

[9] Maiorov V, Meir R S. Approximation bounds for smooth functions in $C(R^d)$ by neural and mixture networks. IEEE Transactions on Neural Networks, 1998, 9(5): 969-978

[10] Cao Fei-Long, Xu Zong-Ben. Neural network approximation for multivariate periodic functions: Estimates on approximation order. Chinese Journal of Computers, 2001, 24(9): 903-908(in Chinese)
(曹飞龙, 徐宗本, 多变元周期函数的神经网络逼近: 逼近阶估计. 计算机学报, 2001, 24(9): 903-908)

[11] Cao Fei-Long, Xu Zong-Ben. Approximation of polynomial functions by neural networks: Construction of network and algorithm of approximation. Chinese Journal of Computers, 2003, 26(8): 906-912(in Chinese)
(曹飞龙, 徐宗本, 梁吉业, 多项式函数的神经网络逼近: 网络的构造与逼近算法. 计算机学报, 2003, 26(8): 906-912)

[12] Timan A F. Theory of Approximation of Functions of a Real Variable. New York: Macmillan, 1963

WANG Jian-Jun, born in 1976, Ph. D., associate professor. His research interests include neural networks, learning theory and approximation theory of functions.

XU Zong-Ben, born in 1955, Ph. D., professor, Ph. D. supervisor. His research interests are in artificial intelligent and nonlinear functional analysis.

Background

Artificial neural networks have been extensively applied in various fields of science and engineering. Various problems concerning application of neural networks in science and engineering can be converted into problems of approximating functions by superposition of the neural activation functions of the networks. The approximation of multivariate functions by the FNNs has been widely studied in past years, with various significant results, concerning density or complexity. However, from the respective of application, the algorithm research on approximation of the neural networks is more hopeful, especially, the determination of topology of neural networks and the parameter of algorithm. Polynomial is a class of fundamental functions. There are many methods to realize its approximation. The authors study an algorithm based on the theory of neural networks, and give theory and experiments of the algorithm. These results reveal the relationship between topology of networks and approximation accuracy in approximation of polynomial.

This research is supported by the National Basic Research Program (973 Program) of China (No. 2007CB311000) and the National Natural Science Foundation of China (Nos. 10726040, 70531030, 10701062, 10826081), the Key Project of Chinese Ministry of Education (No. 108176), China Postdoctoral Science Foundation (No. 20080431237). The group already achieved some research works in the area of theory and application of neural works, which was published in area premium journals such as Science in China Series E: Information Science, Neural Networks, Chinese Journal of Computers and so on. As an essential part of the project, both the theory and experimental results will deepen the research and contribute to above projects. Currently, the authors and their research group are conducting intensive research in this field to get more and better results.