

序列泛函网络模型及其学习算法与应用

周永权^{1),2)} 赵斌³⁾ 焦李成¹⁾

¹⁾(西安电子科技大学智能信息处理研究所 西安 710071)

²⁾(广西民族大学数学与计算机科学学院 南宁 530006)

³⁾(中央民族大学理学院 北京 100081)

摘 要 通过对泛函网络的分析,提出了一种序列泛函网络模型及学习算法,而网络的泛函参数利用梯度下降法来进行学习.在此基础上,给出了9种典型泛函方程对应的序列泛函网络求解模型以及一种基于序列泛函网络学习算法的求解泛函方程方法.通过算例进行仿真实验,结果表明,该方法十分有效,具有收敛速度快、计算精度高、泛化性能好等特点,解决了传统的数值方法难以求解泛函方程这个问题.该方法可用于一般泛函方程求解问题.

关键词 泛函网络; 序列泛函网络; 学习算法; 泛函方程

中图法分类号 TP18

Serial Function Networks Method and Learning Algorithm with Applications

ZHOU Yong-Quan^{1),2)} ZHAO Bin³⁾ JIAO Li-Cheng¹⁾

¹⁾(Institute for Intelligence Information Processing, Xidian University, Xi'an 710071)

²⁾(College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning 530006)

³⁾(School of Science, Central University for Nationalities, Beijing 100081)

Abstract In this paper, by analyzing the functional network, a new model and learning algorithm of the serial functional networks is proposed. And the learning of parameters of the serial functional networks is carried out by the gradient descent algorithm. Based on this, nine kinds of serial function networks for solving classical functional equations and a kind of solving functional equations method on serial functional networks are presented. The simulation results show that the identification method presented in the paper has rapid convergence speed and powerful performance. Contrary to traditional numerical method, this method in this paper could be used to solve arbitrary functional equations.

Keywords functional network; serial functional network; learning algorithm; functional equations

1 引言

Castillo(1998年)提出了泛函网络(functional networks)^[1-2]的概念,在此基础上,1999年又提出了序列泛函网络(serial functional networks)模

型^[3].序列泛函网络是目前应用广泛的泛函网络模型之一,它的泛函表达式是各个输入变量复合作用效果的组合,是对递归人工神经网络的一般化推广和简化.迄今为止,泛函网络已被成功地应用于系统辨识^[2],混沌时间序列的预测,微分、差分 and 泛函方程的求解,CAD,线性、非线性回归^[3-6],计算机代

数^[7-16],混沌通信解码^[17]等;泛函网络在解决上述问题中都表现出较好的性能.虽然泛函网络在应用方面取得一定的成功,由于泛函网络是近几年刚提出的,有些理论和应用方面的基础还不太健全,需要我们不断提出更适用于所要解决问题的新模型,完善基础理论,提出新的算法,更进一步拓展泛函网络的应用范围.

针对目前人们对泛函网络研究的现状,我们基于泛函网络拓扑结构的特点,采用复合函数或映射合成的观点,设计出了一类序列泛函网络模型,提出了一种序列泛函网络学习算法,而网络的泛函参数利用梯度下降法来进行学习.在此基础上,给出了9种求解典型泛函方程序列泛函网络模型以及一种基于序列泛函网络学习算法的求解泛函方程方法.最后,利用序列泛函网络来求解泛函方程并通过一应用实例进行仿真实验,结果表明,该算法十分有效,具有收敛速度快、计算精度高、泛化性能好等特点,较好地解决了传统的数值方法难以求解泛函方程这个问题,对于求解一般泛函方程具有较高的参考价值.

2 序列泛函网络

序列泛函网络^[3]以单层泛函单元(one-layer functional units)序列为基本部件,依据系统的层次特性,借助于函数映射合成的观点,构造相应的泛函网络模型,这样既保留了泛函网络的特性和优点,又能在物理意义上很好地逼近具有迭代结构的问题系统,它的拓扑结构描述的是一个迭代函数变换系统.因此,首先有必要对泛函网络的拓扑结构做一简要介绍.

2.1 泛函网络

泛函网络由以下元素组成:(1)输入单元层.这是输入数据的一层单元.输入单元以带有相应名字的实心圆表示;(2)输出单元层.这是最后一层单元,用于输出网络的结果数据.输出单元也是以带有相应的名字的实心圆表示;(3)一层或多层神经元或计算单元.每一个神经元是一个计算单元,它计算一组来自前一层神经元或输入单元的输入值,并为下一层神经元或输出单元提供一组输入数据.计算单元相互连接,每一个神经元的输出可作为另一个神经元或输出单元数据的一部分,一旦给定输入值,输出便由神经元的类型确定.它由一函数定义.例如,假定我们有一个神经元具有 s 个输入 (x_1, x_2, \dots, x_s) 及 k 个函数 $f_j, j=1, 2, \dots, k$, 使得 $y_j =$

$f_j(x_1, x_2, \dots, x_s); j=1, 2, \dots, k$. 函数 f_j 由网络的结构确定,神经元由带有相应的 f_j 函数名称圆圈来表示;(4)有向连接线.它们将输入层、第一层神经元、第二层神经元、最后一层神经元及输出单元连接起来,箭头表示信息流动的方向,所有这些一起形成网络结构.它确定了网络的泛函能力.网络的结构是指神经元的组织及包含的连接.图1给出一个典型的泛函网络模型.

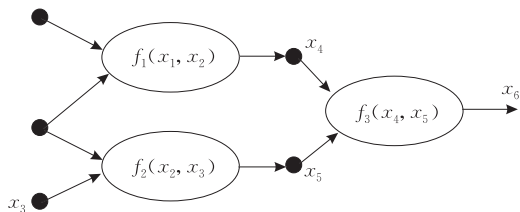


图1 一个泛函网络模型图

从泛函网络模型结构可看出,除了它本身结构特点外,最大的缺陷是该类型网络无反馈功能,这对人们处理具有层次序列特征的问题带来不便,它有可能造成网络的规模急剧地增加,使得系统建模变得更加复杂.因此,考察序列泛函网络结构模型、学习算法和应用,具有重要的理论意义和应用价值.

2.2 序列泛函网络

序列泛函网络 SFN (Serial Functional Networks) 是以单层泛函单元 (one-layer functional units) 序列为基本部件,依据构成该系统的先后次序这个特性,再结合函数映射合成的观点构造出来的泛函网络,我们称之为序列泛函网络(图2).

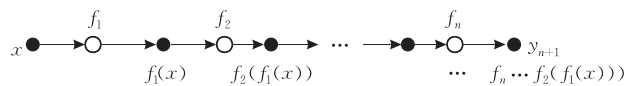


图2 序列泛函网络 SFN (○代表泛函神经元 f_i)

在图2中, f_1, f_2, \dots, f_n 表示 n 个泛函神经元, x 表示输入变量, y_{n+1} 表示输出变量,该网络的输出

$$y_{n+1} = f_n(f_{n-1}(\dots f_1(x)\dots)) \quad (1)$$

特别地,对 $\forall i, f_i = f$, 方程(1)可变为

$$y_{n+1} = f(f(\dots f(x)\dots)) = f^{(n)}(x) \quad (2)$$

根据 Castillo 的做法^[1],对每一泛函神经元 f_i 来说,都可表示成一个已知函数簇的线性组合的形式

$$f_i(x) = \sum_{j=1}^m a_{ij} \varphi_{ij}(x) = \mathbf{a}_i^T \boldsymbol{\varphi}_i(x), i=1, 2, \dots, n \quad (3)$$

其中 $\{\varphi_{ij}(x), i=1, 2, \dots, n, j=1, 2, \dots, n\}$ 是给定的,适合于逼近 f_i 到期望精度的,线性独立的函数

集, $\boldsymbol{\varphi}_i(x) = (\varphi_{i1}(x), \varphi_{i2}(x), \dots, \varphi_{in}(x))^T$, $\mathbf{a}_i = (a_{i1}, a_{i2}, \dots, a_{in})^T$. 系数 a_{ij} 是序列泛函网络的参数.

根据式(3), 对每一泛函神经元 $f_i (i=1, 2, \dots, n)$, 我们有

$$\begin{cases} f_1(x) = \sum_{j=1}^{m_1} a_{1j} \varphi_{1j}(x) = \mathbf{a}_1^T \boldsymbol{\varphi}_1(x) \\ f_2(x) = \sum_{j=1}^{m_2} a_{2j} \varphi_{2j}(x) = \mathbf{a}_2^T \boldsymbol{\varphi}_2(x) \\ \vdots \\ f_n(x) = \sum_{j=1}^{m_n} a_{nj} \varphi_{nj}(x) = \mathbf{a}_n^T \boldsymbol{\varphi}_n(x) \end{cases} \quad (4)$$

再根据式(1), 我们有如下表达式

$$y_{n+1} = \mathbf{a}_n^T \boldsymbol{\varphi}_n(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x))\dots)) \quad (5)$$

从式(5)可看出, 序列泛函网络本质上对应的仍然是一泛函变换序列, 它的拓扑结构描述的是一个迭代函数变换系统.

3 序列泛函网络学习算法

同 ANN 一样, 泛函网络也要学习, 所不同的是泛函网络不是权的学习, 而是结构和参数的学习, 学习的目的就是求出神经元函数的精确表达式或近似表达式. 因而有两种学习方式: 精确学习和近似学习. 精确学习就是确定泛函网络所表示的泛函方程的解函数, 但在实际应用中, 往往不知道泛函方程的精确解, 那么可用一些数据样本近似学习泛函函数. 近似学习就是依据给定样本来评价神经元函数, 其基本方法是线性组合基函数簇的函数, 并且优化线性组合中的系数, 如果函数是线性的, 用相关优化函

数产生一组线性方程组, 评价其系数, 如果函数是非线性的, 存在着多个最优参数, 用梯度下降法迭代出最佳参数. 对于序列泛函网络来说, 对于式(5)函数是线性的情形简单, 本文我们考虑函数是非线性的情形的学习算法.

设误差函数可表示为

$$e_p = d_p - y_p = d_p - \mathbf{a}_n^T \boldsymbol{\varphi}_n(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots)) \quad (6)$$

式中 d_p 为第 p 个训练模式的期望输出. 这样为了找到最优的网络参数, 我们需要最小化下面的误差平方和(P 为当前训练模式数):

$$E_0 = \sum_{p=1}^P e_p^2 = \sum_{p=1}^P [d_p - \mathbf{a}_n^T \boldsymbol{\varphi}_n(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots))]^2 \quad (7)$$

为了保证序列泛函网络表达式的惟一性, 需要给出网络的一些初值. 在这里我们设网络的初始值为

$$f_i(x_{i,0}) = \mathbf{a}_i^T \boldsymbol{\varphi}_i(x_{i,0}) = u_{i,0}, \quad i = 1, 2, \dots, n \quad (8)$$

式中 $u_{i,0}$ 是给定的一些常数, 在许多情况下可能并不需要所有的 n 个初始值, 但为了找到一般的学习算法, 我们还需要假设出所有的初始条件. 这样通过加惩罚项(式(7)中 E_0), 即可得到如下的目标函数

$$E = E_0 + c \sum_{i=1}^n (f_i(x_{i,0}) - u_{i,0})^2 = \sum_{p=1}^P [d_p - \mathbf{a}_n^T \boldsymbol{\varphi}_n(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots))]^2 + c \sum_{i=1}^n (f_i(x_{i,0}) - u_{i,0})^2 \quad (9)$$

式(9)对参数 $\mathbf{a}_i (i=n, n-1, \dots, 1)$ 求偏导得

$$\begin{cases} \frac{\partial E}{\partial \mathbf{a}_n} = -2 \sum_{p=1}^P e_p \boldsymbol{\varphi}_n(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots)) + 2c(f_n(x_{n,0}) - u_{n,0}) \boldsymbol{\varphi}_n(x_{n,0}) \\ \frac{\partial E}{\partial \mathbf{a}_{n-1}} = -2 \sum_{p=1}^P e_p \mathbf{a}_n^T \boldsymbol{\varphi}_n'(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots)) \cdot \boldsymbol{\varphi}_{n-1}(\mathbf{a}_{n-2}^T \boldsymbol{\varphi}_{n-2}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots)) + \\ 2c(f_{n-1}(x_{n-1,0}) - u_{n-1,0}) \boldsymbol{\varphi}_{n-1}(x_{n-1,0}) \\ \frac{\partial E}{\partial \mathbf{a}_{n-2}} = -2 \sum_{p=1}^P e_p \mathbf{a}_n^T \boldsymbol{\varphi}_n'(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}'(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots)) \cdot \boldsymbol{\varphi}_{n-2}(\mathbf{a}_{n-3}^T \boldsymbol{\varphi}_{n-3}(\dots(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots)) + \\ 2c(f_{n-2}(x_{n-2,0}) - u_{n-2,0}) \boldsymbol{\varphi}_{n-2}(x_{n-2,0}) \\ \vdots \\ \frac{\partial E}{\partial \mathbf{a}_1} = -2 \sum_{p=1}^P e_p \mathbf{a}_n^T \boldsymbol{\varphi}_n'(\mathbf{a}_{n-1}^T \boldsymbol{\varphi}_{n-1}'(\dots(\mathbf{a}_2^T \boldsymbol{\varphi}_2'(\mathbf{a}_1^T \boldsymbol{\varphi}_1(x_p))\dots)) \cdot \boldsymbol{\varphi}_1(x_p) + \\ 2c(f_1(x_{1,0}) - u_{1,0}) \boldsymbol{\varphi}_1(x_{1,0}) \end{cases} \quad (10)$$

可以通过下式来自适应地调整泛函网络参数

$$a_i(n+1) = a_i(n) + \frac{1}{2}\mu\left(-\frac{\partial E}{\partial a_i}\right), i = 1, 2, \dots, n$$

(11)

式中, μ 是学习率.

4 基于序列泛函网络的泛函方程求解

泛函方程理论^[18]自 1815 年 Babbage 提出以来, 逐渐发展成为数学的一个分支, 迄今为止, 任意给定一个泛函方程, 在许多情况下, 寻求它的解(如泛函方程 $\varphi(\varphi(x)) = f(x)$)仍然是一件很困难的事情. 针对这类问题, 在本节, 以一些典型的泛函方程为例, 利用序列泛函网络的拓扑结构和学习算法来寻求泛函方程的近似解. 为了以下叙述方便, 首先, 我们先给出一些量的定义, 在此基础上, 设计出一类典型的泛函方程对应的泛函网络求解模型.

设 x 是实数或 n 维实向量, $f(x)$ 是已知函数, $I(x)$ 是恒等函数, $\varphi(x)$ 是未知函数, 它是通过给定的泛函方程所求的函数, c 是任意给定的常数, $(x, f(x))$ 是给定的训练样本数据.

4.1 序列泛函网络模型

4.1.1 逆函数方程

$$\varphi(f(x)) = x$$

(12)

该方程的解为 $\varphi = f^{-1}$. 对应的序列泛函网络见图 3.

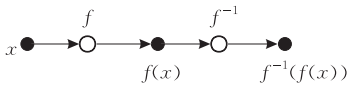


图 3 求解逆函数方程网络结构

从网络的结构可看出, 用泛函网络求解这个方程最简单的方法是将目标数据集和输入训练数据集交换即可完成.

4.1.2 Schröder's 方程

$$\varphi(f(x)) = c\varphi(x)$$

(13)

Schröder's 方程^[19]是最重要的泛函方程之一, 它对于解决其它泛函方程具有指导意义. 若 $\varphi(\cdot)$ 是双射函数, 则该方程可写成

$$f(x) = \varphi^{-1}(c\varphi(x))$$

(14)

对应的泛函网络见图 4.

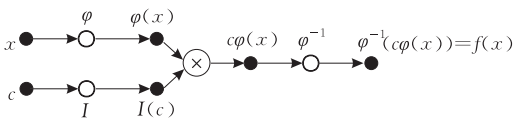


图 4 Schröder's 方程泛函网络模型

其中, $I(\cdot)$ 为恒等泛函神经单元.

4.1.3 Abel's 方程

$$\varphi(f(x)) = \varphi(x) + c$$

(15)

若 $\varphi(\cdot)$ 是双射函数, 则该方程可写成

$$f(x) = \varphi^{-1}(\varphi(x) + c)$$

(16)

对应的泛函网络见图 5.

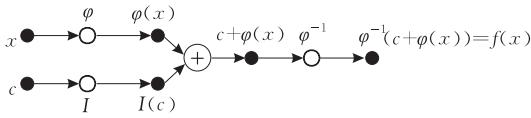


图 5 Abel's 方程泛函网络模型

4.1.4 Iterative Roots 方程

$$\varphi(\varphi(x)) = f(x)$$

(17)

该方程对应的泛函网络见图 6.

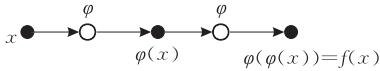


图 6 Iterative Roots 方程泛函网络模型

Iterative Roots 方程本质上是一组迭代函数. 它的逆问题在动力系统、混沌系统和工业处理控制过程中都有着广泛的应用. 特别地, 当 $f(x) = x$ 时, 该方程便是著名的 Babbage 方程.

4.1.5 Fractional 方程

$$\varphi^n(x) = f^m(x)$$

(18)

其中 n, m 是正整数. 该方程对应的泛函网络见图 7.

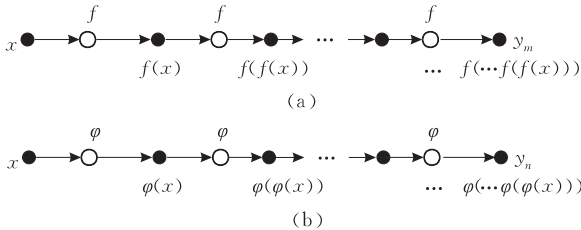


图 7 Fractional 方程泛函网络模型

4.1.6 Commuting Functions 方程

$$\varphi(f(x)) = f(\varphi(x))$$

(19)

该泛函方程所表示的意义是两个函数的次序可交换, 对应地泛函网络的输出保持不变. 对应的泛函网络模型见图 8.

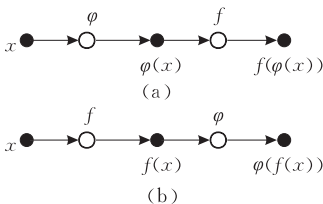


图 8 Commuting Functions 方程泛函网络模型

4.1.7 Periodic 方程

$$\varphi(x) = \varphi(x + c) \tag{20}$$

该方程中常数 c 表示的意义是对输入数据进行放大或缩小,使得函数的值前后不变,它为函数的周期.对应的泛函网络模型见图 9.

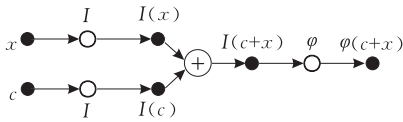


图 9 Periodic 方程泛函网络模型

4.1.8 Gamma 函数方程

$$x\varphi(x) = \varphi(x + 1) \tag{21}$$

对给定的整数,该方程定义 $x!$;对任意一实数 x 来说,该方程的一个解是一个 Euler's Gamma 函数;除此之外,它还有其它的解,对应的泛函网络模型见图 10.

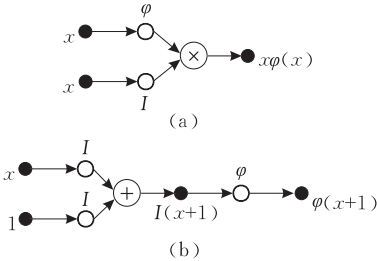


图 10 Gamma 函数方程泛函网络模型

4.1.9 Feigenbaums 方程

$$c\varphi(x) = \varphi(\varphi(cx)) \tag{22}$$

这个方程在动力系统归一化理论等领域有着广泛的应用^[20].该泛函方程对应的泛函网络模型见图 11.

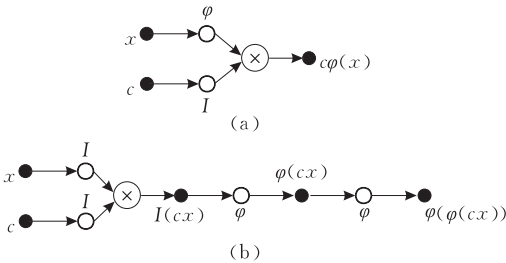


图 11 Feigenbaums 方程泛函网络模型

4.2 表达式唯一性

引理 1^[21]. 平移泛函方程:

$$F[F(x, m), n] = F(x, m + n);$$

$$x, f(x, n) \in (a, b); m, n \in R.$$

若以下条件有一个成立:

- (1) $F(x, n)$ 关于 n 对每一个 x 值和关于 x 对无数个 n , $F(x, n)$ 是严格单调的;
- (2) $F(x, n)$ 关于 x 对每一个 n 值和关于 n 对于 $x = x_0$,即 x 是一固定的值, $F(x, n)$ 是连续的,

则 $F(x, n)$ 有通解:

$$F(x, n) = g^{-1}[g(x) + n],$$

其中 $g(\cdot)$ 是任一严格单调实函数.

由以上引理,我们可看出,函数 f 的 n 次迭代可写成

$$f^{(n)}(x) = F(x, n) = g^{-1}[g(x) + n],$$

特别当 $n=1$ 时,

$$f(x) = F(x, 1) = g^{-1}[g(x) + 1].$$

在泛函网络中一个重要的问题就是表达式唯一性(uniqueness of representation)问题,也就是一个泛函方程有多种表达式时,它们各种表达式之间的关系如何?以下我们以 Abel's 方程(见图 5)、Schröder's 方程(见图 4)为例,考察其网络输出表达式的唯一性.

定理 1. 设泛函方程(15)或(16)存在两个不同函数 $\varphi(x)$ 和 $\gamma(x)$,若满足:

$$\varphi^{-1}[\varphi(x) + c] = \gamma^{-1}[\gamma(x) + c] \tag{23}$$

其中 c, d 是任意常数,则有

$$\varphi(u) = \gamma(u) + d \tag{24}$$

证明. 根据方程(14),我们可写成

$$\gamma\varphi^{-1}[\varphi(x) + c] = \gamma(x) + c; \forall x \tag{25}$$

置 $x = \gamma^{-1}(x)$,我们可得到

$$\gamma\varphi^{-1}[\varphi(\gamma^{-1}(x)) + c] = \gamma(\gamma^{-1}(x)) + c$$

$$= x + c; \forall x,$$

进一步,我们令: $s(x) = \varphi(\gamma^{-1}(x))$,代入上式,有

$$s(x) + c = s(x + c) \tag{26}$$

该方程是一个差分方程,它的一般解为

$$s(x) = x + d \Leftrightarrow \varphi(\gamma^{-1}(x)) = x + d \Leftrightarrow \gamma^{-1}(x)$$

$$= \varphi^{-1}(x + d) \tag{27}$$

式(27)意味着

$$\varphi(u) = \gamma(u) + d,$$

其中 d 是任意常数.

证毕.

同样的方法,我们可考虑 Schröder's 方程表达式的唯一性.

定理 2. 设泛函方程(13)或(14)存在两个不同函数 $\varphi(x)$ 和 $\gamma(x)$,若满足:

$$\varphi^{-1}[c\varphi(x)] = \gamma^{-1}[c\gamma(x)] \tag{28}$$

其中 c, d 是任意常数,则有

$$\varphi(u) = d\gamma(u) \tag{29}$$

证明. 根据方程(13),我们可写成

$$\gamma\varphi^{-1}[c\varphi(x)] = c\gamma(x); \forall x \tag{30}$$

置 $x = \gamma^{-1}(x)$,我们可得到

$$\gamma\varphi^{-1}[c\varphi(\gamma^{-1}(x))] = c\gamma(\gamma^{-1}(x)) = cx; \forall x \tag{31}$$

进一步,我们令: $s(x) = \varphi(\gamma^{-1}(x))$,代入上

式,有

$$cs(x) = s(cx) \tag{32}$$

同样,该方程是一个差分方程,它的一般解为

$$s(x)=dx \Leftrightarrow \varphi(\gamma^{-1}(x))=dx \Leftrightarrow \gamma^{-1}(x)=\varphi^{-1}(dx) \tag{33}$$

这样,我们可得到

$$\varphi(u) = d\gamma(u)$$

其中, d 是任意常数. 证毕.

5 仿真结果及分析

为了验证文中所提出序列泛函网络学习算法的有效性,定义均方误差

$$RMSE = \sqrt{\frac{1}{P} \sum_{i=1}^P \|y_i - \hat{y}_i\|^2} \tag{34}$$

其中, \hat{y}_i 表示网络的输出, y_i 表示网络的实际输出, P 表示学习样本数.

例 1. 用序列泛函网络(图 5)求解 Abel's 方程

$$f(x) = \varphi^{-1}(\varphi(x) + c).$$

其中: $f(x)$ 是任意给定的函数; $\varphi(x)$ 是待确定的函数,即所给泛函方程的解.对于有教师的训练算法来讲,事先给定一组学习样本数据 $\{(x_i, y_i) \mid i = 1, 2, \cdots, n\}$, 此处, $y_i = f(x_i)$.根据泛函方程(9)可得到

$$\varphi(y_i) = \varphi(x_i) + c, \quad \forall i = 1, 2, \cdots, n \tag{35}$$

我们将 $\varphi(\cdot)$ 表示成一个已知函数簇 $\{\gamma_i(x) \mid i = 1, 2, \cdots, m\}$ 的线性组合的形式,即

$$\varphi(x) = \sum_{i=1}^m c_i \gamma_i(x) \tag{36}$$

则误差代价函数为

$$e_i = \varphi(x_i) - \varphi(y_i) + c = \sum_{j=1}^m c_j (\gamma_j(x_i) - \gamma_j(y_i)) + c \tag{37}$$

为了得到泛函参数 c_1, c_2, \cdots, c_m , 我们需要最小化误差平方和

$$E = \sum_{i=1}^n \left(\sum_{j=1}^m c_j (\gamma_j(x_i) - \gamma_j(y_i)) + c \right)^2 \tag{38}$$

为了得到一般求解算法,设泛函网络(见图 5)的初始条件为

$$\varphi(x_0) = \sum_{j=1}^m c_j \gamma_j(x_0) = \alpha \tag{39}$$

其中, α 是给定的常数.这样,可得到如下目标函数

$$E = \sum_{i=1}^n \left(\sum_{j=1}^m c_j (\gamma_j(x_i) - \gamma_j(y_i)) + c \right)^2 + \lambda \left(\sum_{j=1}^m c_j \gamma_j(x_0) - \alpha \right) \tag{40}$$

这样,我们可利用式(7)~(11),完成对泛函网

络的学习,从而得到满意泛函参数 c_1, c_2, \cdots, c_m , 最终可获得所求泛函方程的近似解.

设给定函数

$$f(x) = \log(1 + \exp(x)) \tag{41}$$

我们计算函数 f 的 n 次迭代 $F(x, n) = f^{(n)}(x)$, 根据定理 1 和图 5 的泛函网络学习算法来完成.使得

$$f(x) = \hat{g}^{-1}(\hat{g}(x) + n) \tag{42}$$

为了估计出函数 $\hat{g}(x)$, 可根据已知函数 $f(x)$, 事先给定一组样本数据^[3](表 1).

表 1 样本数据 (x_t, y_t) 用于逼近 $g(x)$

x_t	y_t	x_t	y_t	x_t	y_t
0.681	1.091	0.919	1.255	0.612	1.045
0.156	0.774	0.728	1.122	0.140	0.766
0.167	0.786	0.888	1.233	0.968	1.290
0.444	0.940	0.925	1.259	0.787	1.162
0.126	0.758	0.493	0.969	0.175	0.785
0.579	1.024	0.519	0.986	0.061	0.724
0.883	1.229	0.436	0.935		

其中, $y_t = f(x_t), t = 1, 2, \cdots, 20$. 根据式(23), 当 $n=1$, 我们有

$$g(y_i) = g(x_i) + 1, \quad \forall i = 1, 2, \cdots, n.$$

为方便起见,再设 $\lambda = 0$; 我们根据式(35)~(40), 有

$$E = \sum_{i=1}^{20} \left(\sum_{i=1}^m c_i (y_i^i - x_i^i) - 1 \right)^2 \tag{43}$$

这样,我们可利用式(7)~(11),完成对网络的学习,从而得到满意泛函参数 c_1, c_2, \cdots, c_m , 最终可获得所求泛函 Abel's 方程的近似解.

以下针对这个问题,给定不同的基函数.选用图 5 所示的泛函网络结构,输入个数 $n=2$, 输入样本数据为 $\{x_t, 1\}$, 学习率 $\mu=0.05$. 其中: $t=1, 2, \cdots, 20$. 我们用表 1 中数据训练这个泛函网络,可求出泛函 Abel's 方程的近似解.

(1) 若对图 5 中泛函神经元函数选取是通过基函数线性叠加的方式,当选取基函数 $g(x) = \{\sin x, \sin(2x), \cos x, \cos(2x), 1, x\}$ 时,利用 Mathematica 5.1 编程计算,我们可得到 Abel's 方程的一个近似解:

$$g(x) = 3.86164 \times 10^{12} + 5.81685x + 0.203921\cos x - 0.246107\cos(2x) - 5.61563\sin x + 0.410299\sin(2x).$$

此时的泛函网络的训练误差和训练最大误差分别为

$$\begin{aligned} Train-RMS-error &= 0.000163775, \\ Train-Max-error &= 0.0283076. \end{aligned}$$

为了得到近似模型(42),我们需要求出 g^{-1} . 由

于 $g(x)$ 是单调函数,采用数值分析中著名的二分法^[22],我们需要计算 $f^n(x)=g^{-1}(g(x)+n)$). 例如,当 $n=1$ 时,有

$$f^1(x)=\{1.10876, 0.799342, 0.804801, 0.960681, 0.783563, 1.04353, 1.24508, 1.27034, 1.13973, 1.24862, 1.27429, 0.990148, 1.00626, 0.956063, 1.06443, 0.7912, 1.30518, 1.1791, 0.809709, 0.750443\}.$$

此时的泛函网络的测试误差和测试最大误差分别为

$$\begin{aligned} \text{Test-RMSE-error} &= 0.0206274, \\ \text{Test-Max-error} &= 0.0263312. \end{aligned}$$

图 12 给出泛函方程数值解的精确值与近似值之间误差变化曲线.

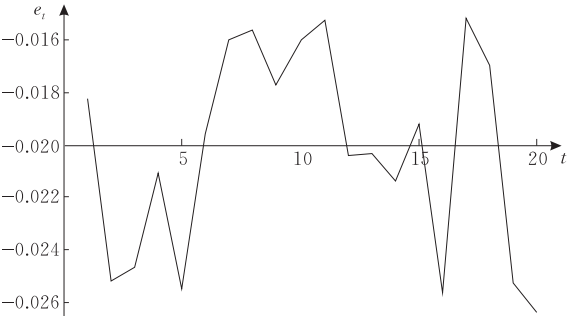


图 12 基函数(1)精确解与近似解之间误差 e_t 变化比较曲线

对应的学习曲线为图 13,而测试曲线为图 14.

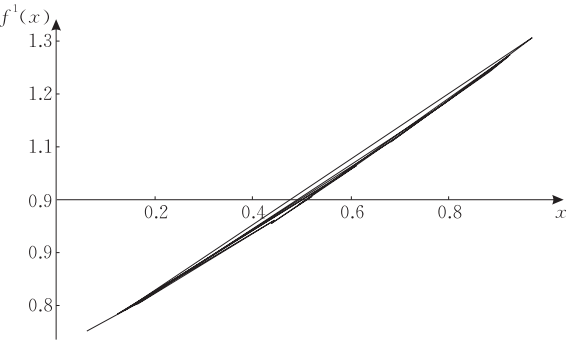


图 13 基函数(1)学习曲线 $f^1(x) \approx g^{-1}(g(x)+1)$

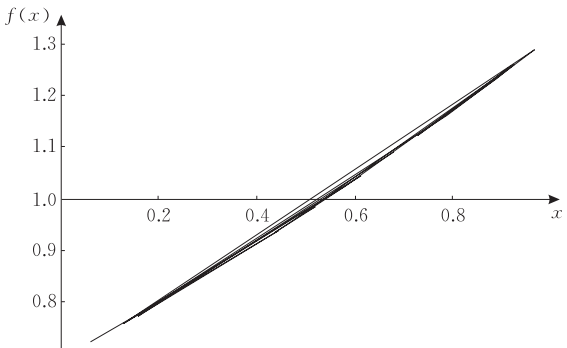


图 14 基函数(1)测试曲线 $f(x)$

(2) 若选取基函数 $g(x)=\{1, \log(1+x), \log(2+x), \log(3+x), \log(4+x), \log(5+x)\}$ 时,利用 Mathematica 5.1 编程计算,我们可得到 Abel's 方程的另一个近似解:

$$\begin{aligned} g(x) &= 3.574 \times 10^{12} + 106.42 \log(1+x) - \\ &\quad 4058.2 \log(2+x) + 26046.7 \log(3+x) - \\ &\quad 50715.7 \log(4+x) + 29602.2 \log(5+x). \end{aligned}$$

此时的泛函网络的训练误差和训练最大误差分别为

$$\begin{aligned} \text{Train-RMS-error} &= 0.000182698, \\ \text{Train-Max-error} &= 0.0127746. \end{aligned}$$

同理,为了得到近似模型(42),我们需要求出 g^{-1} . 由于 $g(x)$ 是单调函数,采用数值分析中著名的二分法,我们需要计算 $f^n(x)=g^{-1}(g(x)+n)$. 例如,当 $n=1$ 时,有

$$\begin{aligned} f^1(x) &= \{1.10751, 0.79736, 0.803536, 0.959201, 0.781774, 1.04222, 1.24402, 1.26921, 1.13843, 1.24755, 1.27329, 0.988633, 1.00453, 0.954698, 1.06286, 0.789277, 1.30417, 1.178, 0.80725, 0.748265\}. \end{aligned}$$

此时,泛函网络的测试误差和测试最大误差分别为

$$\begin{aligned} \text{Test-RMSE-error} &= 0.0191161, \\ \text{Test-Max-error} &= 0.024153. \end{aligned}$$

图 15 给出泛函方程数值解的精确值与近似值之间误差变化曲线.

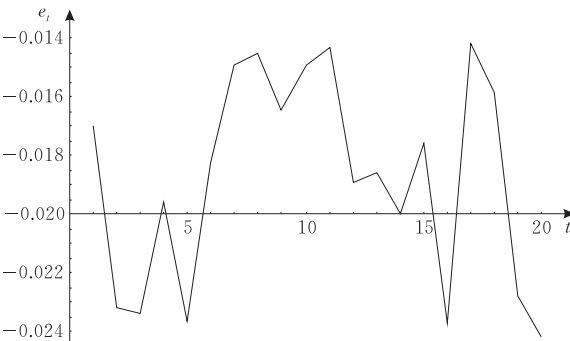


图 15 基函数(2)精确解与近似解之间误差 e_t 变化比较曲线

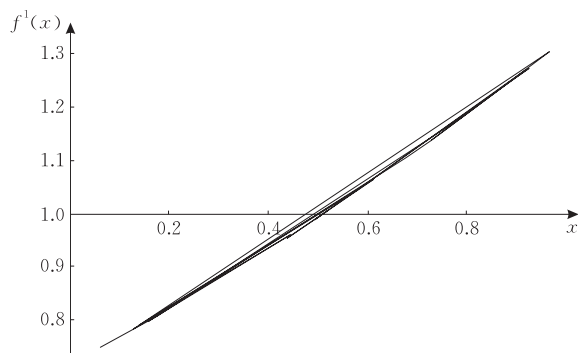
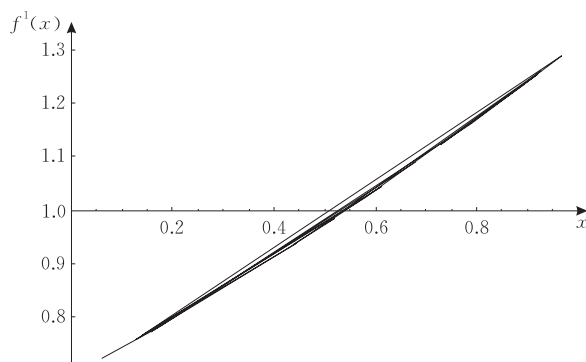
对应地学习曲线为图 16.

比较图 13 和图 16 学习曲线差别,我们可看出,当基函数选取不同,它们之间的差别是微小的.

(3) 若选取基函数 $g(x)=\{x, x^2, x^3, x^4\}$ 时,利用 Mathematica 5.1 编程计算,我们得到 Abel's 方程的另一个近似解^[3]:

$$\begin{aligned} g(x) &= 0.999124x + 0.513156x^2 + \\ &\quad 0.130564x^3 + 0.0754621x^4 \end{aligned}$$

此时,泛函网络的训练误差和训练最大误差分

图 16 基函数(2)学习曲线 $f^1(x) \approx g^{-1}(g(x) + 1)$ 图 18 基函数(3)学习曲线 $f^1(x) \approx g^{-1}(g(x) + 1)$

别为

$$\text{Train-RMS-error} = 0.0001779,$$

$$\text{Train-Max-error} = 0.000198836.$$

采用数值分析中著名的二分法来计算: $f^n(x) = g^{-1}(g(x) + n)$. 例如, 当 $n=1$ 时, 有

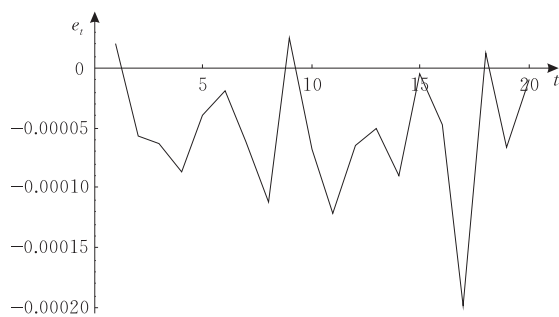
$$f^1(x) = \{1.09051, 0.774243, 0.780192, 0.939675, 0.75817, 1.024, 1.22916, 1.25481, 1.12196, 1.23271, 1.25911, 0.96979, 0.985997, 0.934813, 1.04526, 0.765643, 1.29017, 1.16214, 0.784537, 0.724122\}.$$

此时, 泛函网络的测试误差和测试最大误差分别为

$$\text{Test-RMSE-error} = 0.0000757672,$$

$$\text{Test-Max-error} = 0.000198836.$$

图 17 给出泛函方程数值解的精确值与近似值之间误差变化曲线.

图 17 基函数(3)精确解与近似解之间误差 e_t 变化比较曲线

对应地学习曲线为图 18.

从上面的算例可看出, 针对同一个泛函方程(如 Abel's 方程), 我们可选取不同基函数, 如三角函数、对数函数、多项式函数等, 就可得到泛函方程不同形式的解. 通过仿真可得出, 泛函网络有很快的收敛速度, 泛函方程的解都具有较高的精度. 相比之下, 选取多项式基函数来求解 Abel's 方程, 训练误差小, 学习效果好. 这也许是由于我们预先知道了该系统函数的结构而选取了多项式基函数簇来逼近的

缘故. 但有一点必须明确, 并不是选什么样的函数都可以逼近任意函数, 用于逼近的函数簇的选取直接关系到收敛速度和计算精度, 选取不当甚至导致不收敛或误差过大, 例如在上边的例子中, 若选取基函数簇 $g(x) = \{\sin x, \sin(2x), \cos x, \cos(2x)\}$, 或 $g(x) = \{1, x\}$ 就属于这种情况.

事实上, 在很多时候, 我们并不是对系统一无所知, 可以根据先验知识来恰当地选取基函数簇, 通过对基函数的学习, 求出泛函网络神经元函数的精确表达式或近似表达式, 来达到快速逼近的目的.

6 结 论

序列泛函网络对应的是泛函变换, 它的拓扑结构描述的是一个迭代函数变换系统. 本文通过对序列泛函网络的分析, 提出了 9 种典型的泛函方程对应地泛函方程求解模型, 理论上证明了解的惟一性; 研究了序列泛函网络神经元函数的学习和参数优化算法, 该算法是将对泛函参数学习过程归结为用梯度下降法来进行学习. 通过算例分析表明, 该算法十分有效, 收敛速度快, 计算精度高, 泛化性能好, 可推广到求解一般泛函方程函数解问题.

参 考 文 献

- [1] Castillo E. Functional networks. Neural Processing Letters, 1998, 7: 151-159
- [2] Castillo E, Gutierrez J M. Nonlinear time series modeling and prediction using functional networks. Extracting information masked by chaos. Physics Letters A, 1998, 244: 71-84
- [3] Castillo E, Cobo A, Gutiérrez J M et al. Functional Networks with Applications. Dordrecht: Kluwer Academic Publishers, 1999
- [4] Castillo E, Cobo A, Gutiérrez J M. Working with differential, functional and difference equations using functional net-

- works. *Applies Mathematics Model*, 1999, 23: 89-107
- [5] Castillo E, Gutierrez J M, Cobo A, Castillo C. A minimax method for learning functional networks. *Neural Processing Letters*, 2000, 11(1): 39-49
- [6] Iglesias A, Arcay B, Cotos J M, Taboada J A, Dafonte C. A comparison between functional networks and artificial neural networks for the prediction of fishing catches. *Neural Computer & Applies*, 2004, 13: 24-31
- [7] Zhou Yong-Quan, Jiao Li-Cheng. Interpolation mechanism of functional networks//*Proceedings of the ICANN. Lecture Notes in Computer Science* 3697. Springer-Verlag, 2005: 45-51
- [8] Zhou Yong-Quan, Jiao Li-Cheng. Approximate factorization learning algorithm of multivariate polynomials based on functional networks. *Journal of Information and Computational Science*, 2005, 2(1): 205-210
- [9] Zhou Yong-Quan, He Deng-Xu, Nong Zheng. Application of functional network to solving classification problems//*Proceedings of the World Academy of Science, Engineering and Technology. Enformatika*, 2005, 7: 390-393
- [10] Zhou Yong-Quan, He De-Xu, Jiao Li-Cheng. A neural computation structure for solving functional equations with functional networks. *Journal of Information and Computational Science*, 2005, 2(4): 835-842
- [11] He De-Xu, Nong Zheng, Zhou Yong-Quan. Approximate factorization of univariate polynomials using functional networks. *Journal of Information and Computational Science*, 2005, 2(3): 473-479
- [12] Zhou Yong-Quan, Jiao Li-Cheng. Application of functional networks to solving functional equations//*Proceedings of the 2005 International Conference on Neural Networks & Brain*. Beijing, China, 2005, 2: 1378-1383
- [13] Zhou Yong-Quan, Jiao Li-Cheng. One-variable interpolation function based on functional networks. *International Journal of Information Technology*, 2006, 12(2): 120-129
- [14] Zhou Yong-Quan, Lin Dao-Zhu, Yang Yin-Dong. Functional networks and tunable activation function neural networks//*Proceedings of the 6th World Congress on Control and Automation*. Dalian, China, 2005, 4: 2757-2762
- [15] Zhou Yong-Quan, Wang Dong-Dong, Zhang Min. Designing functional networks through evolutionary programming//*Proceedings of the 6th World Congress on Control and Automation*. Dalian, China, 2005, 4: 3250-3254
- [16] Zhou Yong-Quan, Jiao Li-Cheng. Universal learning algorithm of hierarchical function networks. *Chinese Journal of Computers*, 2005, 28(8): 1277-1286(in Chinese)
(周永权, 焦李成. 层次泛函网络整体学习算法. *计算机学报*, 2005, 28(8): 1277-1286)
- [17] Li Wei-Bin, Liu Fang, Jiao Li-Cheng. Functional network and its application to extract information from chaotic communication. *Journal of System Engineering and Electronics*, 2004, 15(1): 46-49
- [18] Babbage C. An essay towards the calculus of functions. *Philosophical Transaction of the Royal Society London*, 1815, 105: 389-423
- [19] Schröder E. Über iterierte functionen. *Mathematics Annual*, 1871, 3: 296-322
- [20] Briggs K M, Dixon T W, Szekeres G. Analytic solutions of the Cvitanovic-Feigenbaum and Feigenbaum-Kadanoff-Shenker equations. *International Journal of Bifurcation and Chaos*, 1998, 8(2): 347-357
- [21] Aczél. *Lectures on Functional Equations and Their Applications*. New York: Academic Press, 1966
- [22] Chen Gong-Ning, Shen Jia-Ji. *Introduction to Numerical Computations*. Beijing: Beijing Normal University Press, 1988(in Chinese)
(陈公宁, 沈嘉骥编. *计算方法导引*. 北京: 北京师范大学出版社, 1988)
- [23] Kindermann L, Lewandowski A, Protzel P. A framework for solving functional equations with neural networks//*Proceedings of the Neural Information Processing, ICONIP2001*. Shanghai, 2001, 2: 1075-1078



ZHOU Yong-Quan, born in 1962, Ph. D., professor. His current research interests include neural network, computation intelligent.

ZHAO Bin, born in 1964, Ph. D., associate professor. His current research interests include computation theory and applies.

JIAO Li-Cheng, born in 1959, professor, Ph. D. supervisor. His current research interests include nature computation, nonlinear science.

Background

This work is supported by the National Natural Science Foundation of China under grant No. 60461001 and the Natural Science Foundation of Guangxi under grant No. 0542048. Functional network is new network model. It is similar to artificial neural network and is network expression of functional equation. The authors have made a lot of researches on the development of functional network and proposed some new functional network models and learning algorithms which have been applied to the computer algebra.

In this paper, based on serial functional networks, a learning algorithm of the serial functional networks is pro-

posed. And the learning of parameters of the serial functional networks is carried out by the gradient descent algorithm. Based on this, nine kinds of serial function networks for solving some important functional equations and a kind of solving functional equations method on serial functional networks are presented. The simulation results demonstrate that the identification method presented in the paper has rapid convergence speed and powerful performance. Contrary to traditional numerical method, this method in this paper could be used to solve general functional equations.