一类自适应泛函网络循环结构与算法

谢竹诚 周永权
(广西民族大学数学与计算机科学学院 南宁 530006)

摘 要 Banach压缩映射原理不仅在泛函分析中占有举足轻重的地位,同时也是数值分析中求解代数方程、常微分方程解存在唯一性,以及数学分析中积分方程求解的重要理论依据。它是数学和工程计算中最常用的方法之一。基于Banach压缩映射原理,提出一种自适应泛函网络循环结构和算法,通过训练该结构使其逼近于目标函数的不动点。通过算例分析表明,该算法具有计算精度高、收敛速度快等特点。所获结果对于神经计算方法的研究具有参考价值。

关键词 压缩映射原理,泛函网络,循环结构,学习算法
中图法分类号 TP18 **文献标识码** A

Adaptive Functional Networks Loop Structures and Learning Algorithm

XIE Zhu-cheng ZHOU Yong-quan
(College of Math and Computer Science, Guangxi University for Nationalities, Nanning 530006, China)

Abstract The Banach contraction theorem not only plays a vital role in functional analysis, but also is an important theoretical basis for the algebraic equations of numerical analysis, the existence and uniqueness of ordinary differential equations and the integral equation of mathematical analysis. It is one of the most common methods in mathematical and engineering calculations. This paper presented adaptive functional networks loop structures which were designed based on the Banach contraction theorem and the learning algorithm. These structures are used for the approximation of the fixed point of unknown functional relations(mappings) represented by training sets. Finally, the simulation results demonstrate that the structure presented in the paper has high precision and stable. The results obtained in this paper are very important for researching the methods of neural computation.

Keywords Contraction theorem, Functional networks, Loop structures, Learning algorithm

1 引言

1922年,S. Banach提出了一个简单而又非常有用的压缩映射原理——不动点定理[1]。Banach压缩映射原理实际上是对Picard逐次逼近法的抽象表述,是一个典型的代数型不动点定理。它不仅可以判定不动点的存在唯一性,而且还可以构造出一迭代序列,逼近不动点到任意精度。迄今压缩映射原理已在科学计算与工程技术领域的各个方面得到广泛地应用,如线性代数方程、差分和微分方程、控制系统理论和最优化问题等。而自适应循环结构是一种既有前馈通路,又有反馈通路的神经网络结构,反馈通路的引入,使得神经网络能够有效地处理与时间序列相关的上下文信息,压缩映射原理为构建该神经网络结构提供了数学理论基础。于是,Marcelo Malini Lamego 2001年提出了训练循环神经网络结构的一般方法和技术[2],并且扩展到多层循环神经网络。但该网络结构较复杂,待学习的参数较多且计算精度较低。而泛函网络与神经网络相比,神经元选择更灵活,学习算法简单,主要是由于该模型是基于"问题驱动"建模来求解,而不是像神经网络那样,由"模型驱动"建模求解。

1998年,西班牙学者Enrique Castillo提出了泛函网络,随后人们对其不断地进行完善和发展,并应用于许多方面,如多元线性回归、半参数非线性回归等[3-9]。2003年,Alfonso Iglesias Nuno等[10,11]通过大量的实验把泛函网络与神经网络的性能进行比较研究,得出了泛函网络的性能优于神经网络。2004年,Changhua Zhu等人[12]把泛函网络应用于网络动态延迟问题,2007年,Han-Bing Qu提出了一种基于贝叶斯理论的广义联接泛函网络[13];国内周永权等对泛函网络在计算机代数、函数逼近等做了许多工作[14]。目前泛函网络也已被成功地应用于许多领域。归结起来,包括模式识别、抗震强度预测、分类问题、特征选择、三维曲线曲面的拟合[15-19]等。相比神经网络,泛函网络在以上领域都表现出了良好的性能。虽然泛函网络在上述应用方面都取得了一定的成功,但有些理论和应用方面的基础不太健全,还需要不断提出更适合于所要解决问题的新模型,完善基础理论,提出新的泛函网络算法,更进一步拓展泛函网络的应用范围。本文基于Banach压缩映射原理,提出一种自适应泛函网络循环结构和学习算法,

到稿日期:2008-11-11 返修日期:2009-02-10 本文受国家自然科学基金(60461001),广西自然科学基金(0832082)和广西民族大学研究生教育创新(gxun-chx0886)资助。
谢竹诚(1983—),男,硕士,主要研究方向为计算智能及其应用;周永权(1962—),男,博士,教授,主要研究方向为计算智能、神经网络及应用,E-mail:yongquanzhou@126.com。

通过训练该结构使其逼近于目标函数的不动点。通过算例分析表明，该算法具有计算精度高、收敛速度快等特点。本文的结果对于神经计算方法的研究具有重要意义。

2 压缩映射原理与自适应循环结构

2.1 压缩映射原理

定义 1(压缩映射) 称 $T:(\Omega,\rho)\to(\Omega,\rho)$是一个压缩映射，如果存在 $0<\alpha<1$，使得：

$$\rho(Tx,Ty)\leqslant\alpha\rho(x,y),\forall x,y\in\Omega$$

定理 1(Banach 压缩映射原理) 设(Ω,ρ)是一个完备的距离空间，T 是(Ω,ρ)到自身的一个压缩映射，则 T 在 Ω 上存在唯一的不动点。

引理 1[2] 对于赋范向量空间 R^n 和 R^q，有子集 $U\subset R^q$ 和闭子集 $X\subset R^n$。考虑连续函数 $f:X\times U\to X$，且 $v=f(x,u)$。对于任意 $u=u_*\in U,x\in X$，如果 f 是一压缩映射，从而根据压缩映射原理，存在唯一一个不动点 x_*，使：

$$x_*=f(x_*,u_*) \tag{1}$$

因 x_* 与给出的 u_* 一一对应，从而存在一映射 $g:U\to X$ 使：

$$x_*=g(u_*) \tag{2}$$

引理 1 的证明可通过上面的定义 1 和定理 1 得到。

实质上，因 f 是压缩映射，据压缩映射原理知，实际上式(1)和式(2)是等价的。因此，由式(1)定义的代数循环结构可由式(2)表示，反之如此。

2.2 基于函数逼近的自适应循环泛函网络结构

一般函数逼近问题可以描述为：如何确定一个函数集合 Φ，使得特定空间 A 上的任意函数 f 可由 Φ 中函数近似地构造出来。存在的基本问题是：(1)如何选择 Φ；(2)如何基于 Φ 中函数构造出近似函数，使得与目标函数有尽可能小的偏差。对函数逼近技术的研究，人们已提出了许多成熟的逼近方法，如代数多项式逼近、三角多项式逼近、插值方法逼近、样条函数逼近等。通常经典的方法是人们采用回归分析的方法，即已知采用样本数据集合$\{(x_i,y_i)\mid x_i\in R^n,y_i\in R,i\in N\}$，构造满足采样数据点误差等约束条件的函数，但存在着相关却不同的领域问题，如在控制等领域，需要求解特定模型，而该模型是否适当，只有当模拟求解该模型后依据其适合性才能评判。回归分析并不适合于求解该类问题，其根本原因在于所采用的样本数据来源往往受到专家领域知识的限制，导致所求解特定模型与实际模型出现偏差，甚至产生错误的结果。幸运的是机器学习方法的发展为解决该问题提供了一条新的思路，它们不仅可以用于数据回归，而且可用于自适应建模，无论是从函数逼近的方法，还是工程逼近方法，这些都拓广了逼近技术的应用领域。

本节将自适应泛函网络循环的拓扑结构特点与函数逼近技术有机地结合起来，给出的基于泛函网络的函数逼近概念如下：如何确定一个函数集合 Φ，使得特定空间 A 上的任意函数 f 可由 Φ 中函数近似地构造出来。具体地讲，给定一个基函数簇 $\Phi=\{g_\lambda(x),\lambda\in\Lambda\}$，其中 Λ 是指标集合，逼近一个函数 $f(x)$的实质是在基函数簇中找出有限的基函数使得 $f(x)$可以精确或比较精确地表示为这些基函数的线性叠加，即 $f(x)=\sum_{i=1}^{N}w_ig_{\lambda_i}(x)$或者 $f(x)\approx\sum_{i=1}^{N}w_ig_{\lambda_i}(x)$。这里 $f(x)$可以是一维函数，也可以是多维函数；对应的泛函网络可以是单输出，也可以是多输出。

在 $f(x)$给定的情形下，考察逼近误差，用均方误差(RMSE)来评价每一次的逼近性能的好坏。通过调整基函数来实现网络的训练，但是，需要注意的是一般根据 $f(x)$的性质，作为学习的“先验知识”的导向，选择合适的基函数簇，使得函数 $f(x)$在函数簇 Φ 上能高效地逼近，这样避免了逼近的盲目性，以达到最佳逼近效果。

综合上面的分析，对于函数逼近问题，若采用自适应循环泛函网络逼近结构，则可描述为由一个未知的映射 $T:U\to Y$，一个训练样本集：

$$S=\{(u^{(i)},y^{(i)}):u^{(i)}\in U,y^{(i)}\in Y,i=1,2,\cdots,I\} \tag{3}$$

其中，$U\subset R^q,Y\subset R^m,y=T(u)$。即用一自适应循环结构寻找一个逼近于未知函数的映射：$\hat{T}:U\to Y$，且满足：

$$y=\hat{T}(u,w) \tag{4}$$

其中，$w\in W\subset R^p$ 是在参数所在的向量。

一般地，自适应参数循环结构定义如下：

$$\begin{cases}x=f(x,u,a)\\ \hat{y}=h(x,u,b)\end{cases} \tag{5}$$

其中：$f:X\times U\to X,h:X\times U\to X,a\in A\subset R^{p_a}$ 和 $b\in B\subset R^{p_b}$ 是要调整的参数，x 是式(5)中的隐变量。

假设式(5)中的第一式中 f 是一压缩映射，则满足引理 1，即存在一个映射 g，从而由式(5)中的第二式得到$\hat{y}$与 u 和参数(a,b)之间的函数关系。这就可实现前面提到的逼近问题。因为式(5)中定义的自适应循环结构与式(4)定义的是等价的，它们之间的关系如图 1 所示。由于 f 是一压缩映射，自适应循环结构可以看作一个迭代过程，这个过程是一个逐次逼近于函数不动点的过程。则由 2.1 节定义 1 可知，任意给定一个初始值 x_0，可以由式(5)中的第一式得到序列 $x_0,x_1,x_2,\cdots$，由压缩映射定义可表示为一个递归形式：

$$x_{k+1}=f(x_k,u,a),k=0,1,2,\cdots\cdots \tag{6}$$

固定的 u，迭代序列 $x_0,x_1,x_2,\cdots$都会收敛于一个不动点 x_*，则 x_* 是式(5)的唯一的解。关于式(6)的收敛性证明可用 Lyapunov 稳定性理论，具体可参看文献[20]。由式(6)得到的序列一般收敛于式(5)的固定点 x_*。

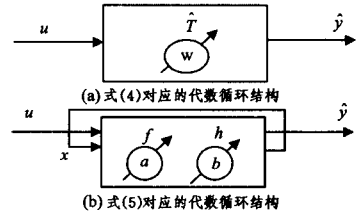

图 1

3 一类自适应泛函网络循环结构

1999 年，Castillo[2] 提出了序列泛函网络 SFN(Serial Functional Networks)，它是以单层泛函单元(one-layer functional units)序列为基本部件，依据系统的先后次序这个特性，结合函数映射合成的观点构造的泛函网络。2008 年，周永权[21]利用泛函网络拓扑结构的特点和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，并提出了相应的算法，其拓扑结构如图 2 所示，其中 $F_1,F_2,\cdots,F_n$ 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，则该网络

的输出为：

$$y_{n+1}=F_n(F_{n-1}(F_{n-2}(\cdots F_1(x))) \tag{7}$$

在特殊情况下，$\forall i F_i=f$，则式(7)变为：

$$y_{n+1}=f(f(f(\cdots f(x)))=f^{(n)}(x) \tag{8}$$

其中，$f^{(n)}(x,u)$表示对 f 迭代 n 次后的值。

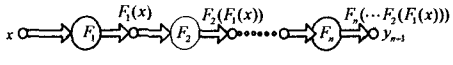

图 2　序列泛函网络结构

由 2.2 节可知，式(5)定义的自适应循环结构中的第一式可表示为一个递归形式，即：

$$x_n=f(f(\cdots f(x_0,u))) \tag{9}$$

则可根据式(8)和式(5)定义的自适应循环结构中的第一式由序列泛函网络得到。因而得出自适应泛函网络循环结构如图 3 所示，且表达式定义如下：

$$\begin{cases} x=f^{(n)}(x_0,u) \\ \wedge \\ y=H(x,u) \end{cases} \tag{10}$$

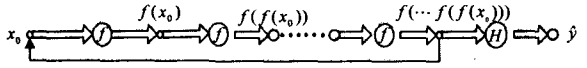

图 3　适应泛函网络循环结构

3.1　自适应泛函网络循环结构的性质

引理 2[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：

(a)$F(x,n)$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。

(b)$F(x,n)$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：

$$F(x,n)=g^{-1}[g(x)+n] \tag{11}$$

其中，$g(\cdot)$是任意一严格单调函数。

由以上引理，则函数 f 的 n 次迭代可写成：

$$f^{(n)}(x,u)=F(x,n)=g^{-1}[g(x,u)+n] \tag{12}$$

特别当 $n=1$ 时，

$$f(x)=F(x,1)=g^{-1}[g(x)+1] \tag{13}$$

则根据引理 2，结合图 3 可以把自适应泛函网络循环结构进行简化，得到一般的自适应泛函网络循环结构的拓扑结构，如图 4 所示，图中的 $f_3=g$，这种结构利于泛函参数的学习。

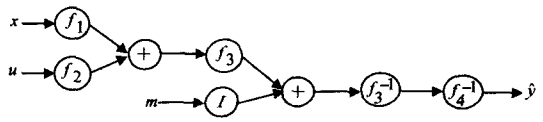

图 4　自适应泛函网络循环结构

3.2　自适应泛函网络循环结构学习算法

根据 Castillo 的做法，对每个泛函神经元 f_i 有：

$$f_i(x)=\sum_{j=1}^{m}a_{ij}\varphi_{ij}(x)=a_i^T\varphi_i(x),i=1,2,\cdots,n \tag{14}$$

其中，$\{\varphi_{ij}(x),i=1,2,\cdots,n,j=1,2,\cdots,n\}$是给定的，适合于逼近 f_i 到期望的精度的、线性独立的函数集，置：

$$\varphi_i(x)=(\varphi_{i1}(x),\varphi_{i2}(x),\cdots,\varphi_{in}(x))^T$$

$$a_i=(a_{i1},a_{i2},\cdots,a_{in})^T$$

系数 a_{ij} 是网络的泛函参数。

对每一泛函神经元 f_i，有：

$$\begin{aligned} f_1(x)&=\sum_{j=1}^{m_1}a_{1j}\varphi_{1j}(x)=a_1^T\varphi_1(x) \\ f_2(x)&=\sum_{j=1}^{m_2}a_{2j}\varphi_{2j}(x)=a_2^T\varphi_2(x) \\ &\cdots\quad\cdots\quad\cdots \\ f_n(x)&=\sum_{j=1}^{m_n}a_{nj}\varphi_{nj}(x)=a_n^T\varphi_n(x) \end{aligned} \tag{15}$$

根据图 4，则 $n=4$ 且有：

$$y=f_4^{-1}[f_3^{-1}[f_3[f_1(x)+f_2(u)]+m]]] \tag{16}$$

即：

$$f_3[f_4(y)]=f_3[f_1(x)+f_2(u)]+m \tag{17}$$

设误差函数为：

$$\begin{aligned} e_p&=f_3[f_4(y_p)]-f_3[f_1(x_p)+f_2(u_p)]-m \\ &=a_3^T\varphi_3[a_4^T\varphi_4(y_p)]-a_3^T\varphi_3[a_1^T\varphi_1(x_p)+a_2^T\varphi_2(u_p)]-m \end{aligned} \tag{18}$$

其中，P 为当前训练模式数。为了找到最优的网络参数，需要最小化误差平方和：

$$\begin{aligned} E_0=\sum_{p=1}^{P}e_p^2=\sum_{p=1}^{P}[f_3[f_4(y_p)]-f_3[f_1(x_p)+f_2(u_p)]-m]^2 \\ =\sum_{p=1}^{P}[a_3^T\varphi_3[a_4^T\varphi_4(y_p)]-a_3^T\varphi_3[a_1^T\varphi_1(x_p)+a_2^T\varphi_2(u_p)]-m]^2 \end{aligned} \tag{19}$$

并为了保证序列泛函网络表达式的唯一性，需要给出网络的一些初值。在这设网络的初始值为：

$$f_i(x_{i0})=a_i^T\varphi_i(x_{i0})=u_{i0} \tag{20}$$

式(20)中 u_{i0} 是给定的一些常数，在许多情况下，并不需要 n 个初始值，但为了找到一般的学习算法，还需要假设给出所有的初始条件。这样通过加惩罚项式(19)中的 E_0。

$$\begin{aligned} E=E_0+\sum_{i=1}^{4}c_i(f_i(x_{i0})-u_{i0})^2=\sum_{p=1}^{P}[a_3^T\varphi_3[a_4^T\varphi_4(y_p)]- \\ a_3^T\varphi_3[a_1^T\varphi_1(x_p)+a_2^T\varphi_2(u_p)]-m]^2+\sum_{i=1}^{4}c_i(f_i(x_{i0})- \\ u_{i0})^2 \end{aligned} \tag{21}$$

式(21)对 a_i 求偏导得：

$$\begin{aligned} \frac{\partial E}{\partial a_4}&=2\sum_{p=1}^{P}e_pa_3^T\varphi'_3(a_4^T\varphi_4(y_p))\cdot\varphi_4(y_p)+2c_4(f_4(x_{40})- \\ &\quad u_{40})\varphi_4(x_{40}) \\ \frac{\partial E}{\partial a_3}&=2\sum_{p=1}^{P}e_p(\varphi_3(a_4^T\varphi_4(y_p))-\varphi_3(a_1^T\varphi_1(x_p)+a_2^T\varphi_2(u_p)))+ \\ &\quad 2c_3(f_3(x_{30})-u_{30})\varphi_3(x_{30}) \\ \frac{\partial E}{\partial a_2}&=-2\sum_{p=1}^{P}e_pa_3^T\varphi_3'(a_1^T\varphi_1'(x_p)+a_2^T\varphi_2'(u_p))\cdot\varphi_2(u_p)+ \\ &\quad 2c_2(f_2(x_{20})-u_{20})\varphi_2(x_{20}) \\ \frac{\partial E}{\partial a_1}&=-2\sum_{p=1}^{P}e_pa_3^T\varphi_3'(a_1^T\varphi_1(x_p)+a_2^T\varphi_2(u_p))\cdot\varphi_1(u_p)+ \\ &\quad 2c_1(f_1(x_{10})-u_{10})\varphi_1(x_{10}) \\ \frac{\partial E}{\partial c_i}&=(f_i(x_{i0})-u_{i0})^2 \end{aligned} \tag{22}$$

令式(22)中的每个等式都等于 0，则得到一个方程组，求解方程组后，就可得到最优的网络参数 a_{ij}。

4　仿真结果及分析

为了验证自适应泛函网络循环结构算法的有效性，用其来求解下列复合泛函方程[22]：

$$f(x-f(y))=f(x)+f(f(y))-axf(y)-bf(y)-c \tag{23}$$

其中，$x,y\in R,a,b,c$ 为 3 个实数。在此，为计算方便，令 $a=1,b=1,c=1$。将式(23)变为：

$$f(x-f(y))=f(x)+f(f(y))-xf(y)-f(y)-1 \quad (24)$$

从而，式(24)可转化为本文提出的自适应泛函网络循环结构的形式：

$$\begin{cases} z=f(x-f^{-1}(z))+xf^{-1}(z)+f^{-1}(z)+f(x)-x^2-x-1 \\ y=f^{-1}(x-f^{-1}(z)) \end{cases} \quad (25)$$

根据上述的复合泛函方程的求解，可对图 4 中的自适应量泛函网络循环结构稍做修改，如图 5 所示，即得到对应的自适应泛函网络循环结构的泛函方程为：

$$\begin{cases} z=f_3^{-1}[f_3[f_1(x)+f_2(z)]+m] \\ y=f_1^{-1}[x-f_1^{-1}(z)] \end{cases} \quad (26)$$

其中，f_3^{-1},f_1^{-1} 分别表示对应函数的逆函数，m 是迭代次数，在这里令 $m=10$。为了计算上述泛函方程，随机选取 20 个数据对来训练网络以达到逼近 $f(x)$效果，训练数据对如表 1 所列。

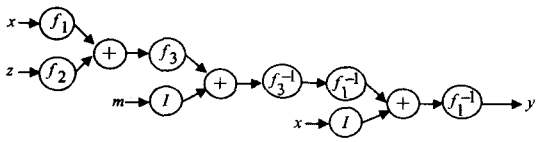

图 5　解复合泛函方程的自适应泛函网络循环结构

表 1　用于逼近 $f(x)$的数据对(x,y)

x	y	x	y	x	y	x	y
0.3796	0.2722	0.9866	0.8506	0.1131	0.8620	0.7627	0.3733
0.2764	0.4194	0.5029	0.3402	0.8121	0.6566	0.7218	0.5314
0.7709	0.2130	0.9477	0.4662	0.9083	0.8912	0.6516	0.1813
0.3139	0.0356	0.8280	0.9138	0.1564	0.4881	0.7540	0.5019
0.6382	0.0812	0.9176	0.2286	0.1221	0.9926	0.6632	0.4222

若取基函数簇：

$(\varphi_{11},\varphi_{12},\varphi_{13})=(1,x,x^2)$，

$(\varphi_{i1},\varphi_{i2})=(1,x),i=2,3$

则为了估计泛函参数$\{a_{ij},i=1,2,3,4;j=1,2\}$，由 3.2 节提出的学习算法，求下式的最小值。

$$\begin{aligned} Q_c=&\sum_{t=1}^{20}(f_3(f_1(x_t-f_1(y_t))-f_3(f_1(x_t)+f_2(z_0))-m)^2+c_1(f_1(x_{10})-u_{10})^2+\sum_{i=2}^{3}c_i(f_i(x_{i0})-u_{i0})^2 \\ =&\sum_{t=0}^{20}(\sum_{i=1}^{2}a_{3i}\varphi_{3i}(\sum_{j=1}^{3}a_{1j}\varphi_{1j}(x_t-\sum_{j=1}^{3}a_{1j}\varphi_{1j}(y_t)))-\sum_{i=1}^{2}a_{3i}\varphi_{3i}(\sum_{j=1}^{3}a_{1j}\varphi_{1j}(x_t)+\sum_{j=1}^{2}a_{2j}\varphi_{2j}(z_0))-m)^2+c_1(\sum_{j=1}^{3}a_{1j}\varphi_{1j}(x_{i0})-u_{i0})^2+\sum_{i=2}^{3}c_i(\sum_{j=1}^{2}a_{ij}\varphi_{ij}(x_{i0})-u_{i0})^2 \end{aligned} \quad (27)$$

其中，$x_{i0},u_{i0},i=1,2,3$ 和 z_0 为任意常数。利用 Mathematica 6 编程计算，求得的泛函参数的结果如表 2 所列。

表 2　泛函参数的计算结果

a_{11}	1.04728×10^{-7}	0.417093	$-0.56487-1.9625i$	$0.11727+0.60415i$
a_{12}	-4.99999×10^{-7}	−1.9974	$2.37310+8.73447i$	$2.89937-4.53568i$
a_{13}	4.40896×10^{-7}	2.2698	$-1.69380-7.92129i$	$-3.12838+5.37851i$
a_{21}	−76.9407	−0.785756	$2.23428+6.03556i$	$-0.74434+2.28171i$
a_{22}	1.31904×10^{-15}	1.83509	$-1.41515-6.49561i$	$3.00116-4.13543i$
a_{31}	0.158304	0.156352	0.156352	0.30126
a_{32}	0.12997	0	0	0
c_1	0	0	0	0
c_2	0	0	0	0
c_3	0	0	0	0
Q_c	3.27014×10^{-29}	1304.98	2000	2000

从表 2 中的计算结果为 $Q_c=3.27014\times10^{-29}$。此时，对应的泛函网络模型式(26)中各神经元函数为：

$f_1(x)=1.04728\times10^{-7}-4.99999\times10^{-7}x+4.40896\times10^{-7}x^2$

$f_2(x)=-76.9407+1.31904\times10^{-15}x$

$f_3(x)=0.158304+0.12997x$

对应式(24)的复合泛函方程的近似解为：

$f(t)=1.104728+0.499999t+0.408960t^2$

与精确解[22] $f(t)=1+0.5t+0.5t^2$ 相近。若选取其它的基函数，同样可获得该方程的近似解。图 6 给出了泛函方程数值解的精确值与自适应循环结构求得的近似值之间误差变化曲线。

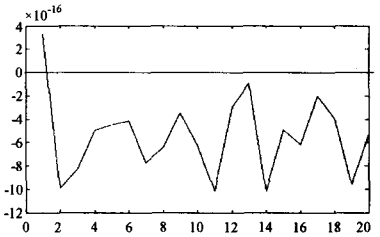

图 6　泛函方程精确值与本文算法的近似值之间误差变化曲线

从图 6 中也可以看出，该模型误差变化的级数都在 10^{-16} 之上，其逼近能力高，所求解是有效的、可行的。

结束语　文中基于 S. Bnach 的压缩映射原理，设计出一类自适应泛函网络循环结构和学习算法，用该算法借助于 Lagrange 乘数法，作辅助函数对泛函参数进行学习。通过算例表明，该算法是可行的，具有算法简单、计算精度高、易于实现等特点。本文的结果对于神经计算方法的研究具有重要意义。

参 考 文 献

[1] Banach S. Sur Les operations dansles ensembles abstraits et leurs applications [J], Fund. Math, 1922(3): 133-181

[2] Lamego M M. Adaptive structures with algebraic loops [J]. IEEE Transactions on Neural Networks, 2001, 12(1): 33-42

[3] Castillo E. Functional Networks [J]. Neural Processing Letters, 1998(7): 151-159

[4] Castillo E, Cobo A, Gutierrez J M. Functional Networks with Applications [M]. Kluwer Academic Publishers, 1999

[5] Castillo E, Gutiérrez J M, Cobo A, et al. A Minimax Method for Learning Functional Networks [J]. Neural Processing Letters (NPL), 2000, 11(1): 39-49

[6] Castillo E, Cobo A, Gómez-Nesterkin R, et al. A general framework for functional networks [J]. Networks, 2000, 35(1): 70-82

[7] Castillo E, Hadi A S, Lacruz B. Optimal Transformations in Multiple Linear Regression Using Functional Networks [J]. IWANN, 2001: 316-324

[8] Castillo E, Fontenla-Romero O, Guijarro-Berdiñas B, et al. A Measure of Noise Immunity for Functional Networks [J]. IWANN, 2001: 293-300

[9] Castillo E, Hadi A S, Lacruz B, et al. Semi-parametric nonlinear regression and transformation using functional networks [J]. Computational Statistics & Data Analysis (CSDA), 2008, 52(4): 2129-2157

[10] Nuno A I, Varela B A, Cotos J M, et al. A Comparison between Functional Networks and Artificial Neural Networks for the Prediction of Fishing Catches[J]. Neural Comput & Applic, 2004, 13: 24-31

[11] Nuno A I, Varela B A, Cotos J M, et al. Optimisation of fishing predictions by means of artificial neural networks, anfis, functional networks and remote sensing images [J]. Expert Syst. Appl. (ESWA), 2005, 29(2): 356-363

[12] Zhu Changhua, Pei Changxing, Li Jiandong. Functional Networks Based Internet End-to-End Delay Dynamics [J]. AINA, 2004: 540-544

[13] Qu Han-bing, Hu Bao-gang. Variational Bayes Inference for Generalized Associative Functional Networks[J]. IJCNN, 2007: 184-189

[14] 周永权. 泛函网络理论及其学习算法研究[D]. 西安: 西安电子科技大学, 2006

[15] Pruneda R E, Lacruz B, Solares C. A First Approach to Solve Classification Problems Based on Functional Networks [J]. ICANN, 2005: 313-318

[16] Sánchez-Maroño N, Caamaño-Fernández M, Castillo E, et al. Functional Networks and Analysis of Variance for Feature Selection [J]. IDEAL, 2006: 1031-1038

[17] Gálvez A, Iglesias A, Cobo A, et al. Bézier Curve and Surface Fitting of 3D Point Clouds Through Genetic Algorithms, Functional Networks and Least-Squares Approximation [J]. ICCSA, 2007: 680-693

[18] El-Sebakhy E A. Functional Networks Training Algorithm for Statistical Pattern Recognition [J]. ISCC, 2004: 92-97

[19] Alonso-Betanzos A, Castillo E, Fontenla-Romero O, et al. Shear Strength Prediction Using Dimensional Analysis And Functional Networks [J]. ESANN, 2004: 251-256

[20] Sastry S. Nonliner Systems: Analysis, Stability and Control[M]. New York: Springer-Verlag, 1999

[21] 周永权, 赵斌, 焦李成. 序列泛函网络模型及其学习算法与应用[J]. 计算机学报, 2008, 31(7): 1073-1091

[22] Akkouchi M. A Class Of Functional Equations Characterizing Polynomials of Degree Two[J]. Divulgaciones, 2001, 9(2): 149-153

(上接第 178 页)

3.2 项目计划动态调整

项目计划的调整分为属性调整和非属性调整。前者是指计划任务的结构没有变更，调整的仅仅是计划中任务的属性和资源的属性，主要包括：

1)项目工期；

2)任务的持续时间；

3)资源的成本；

4)资源的数量。

非属性调整则是指任务或者资源的增加或删除，或者任务层次结构、先后顺序的变更。

显然，项目计划的属性调整与过程参数配置的变化是相互对应的，而非属性调整则对应于过程结构的变化。

同仿真优化一样，项目计划的调整从时间上也分为两个阶段：

1)项目实施前的调整；

2)项目实施过程中的调整。

从调整的原因上又可分为主动调整与被动调整。前者是指项目决策人员根据需求变化需要对当前项目的相关信息进行调整，此时需要主动对过程模型进行修改和优化，之后重新生成项目计划。这种调整的变动一般是比较大的。

被动调整则是指当项目的实际运行状态与计划发生偏离后，过程仿真优化通过事中分析重新生成了新的过程模型，以此引起的后期项目计划的变更。

由于项目实施前的调整不涉及实际运行数据，因此动态调整主要是指在项目实施阶段根据实际情况及时做出的各种主、被动调整。

结束语 综上所述，过程管理是支持企业产品生命周期管理的重要技术，项目管理是产品生命周期高质量完成的保障。本文提出的集成框架结构，为实现两者的信息共享和优化奠定了基础。以动态优化技术为核心的优化循环模型通过过程管理使得项目管理更加科学，通过项目管理为过程实施提供保证。由于事中分析应用了项目实施过程中采集的数据，因此由此得出的过程模型更加贴近企业的实际运行情况，不但为企业决策层提供了及时的分析结果，而且为动态、及时地进行项目计划调整提供了支持，有助于提高项目管理和过程管理的有效性。

参 考 文 献

[1] 周永华, 陈禹六, 赵天奇. 经营过程建模[J]. 计算机集成制造系统, 2002, 8(1): 16-22

[2] 谭文安. 企业过程动态优化技术及其支持环境的研究与开发[D]. 北京: 北京航空航天大学, 2001

[3] Lawrance M, Chung L, Chan K C C. Integrating Project Planning and Process Modeling for Software Development[C]// Proc of the IEEE Symposium on Application-Specific Systems and Software Engineering and Technology. 1999: 276

[4] 范玉顺. 集成化企业建模方法与系统[M]. 北京: 中国电力出版社, 2007

[5] April J, Better M, Glover F, et al. Enhancing Business Process Management with Simulation Optimization[R]. BP Trends Newsletter, White Paper & Technical Briefs, 2005: 1-11

[6] 孙雪冬, 徐晓飞, 王刚. 基于有向超图的资源约束下企业过程结构优化[J]. 软件学报, 2006, 17(1): 59-67

[7] 姚咏, 范文慧, 熊光楞. 复杂产品开发过程仿真及其优化方法研究[J]. 系统仿真学报, 2006, 18(3): 726-730

[8] Dewan R, Seidmann A, Walter Z P. Workflow optimization through task redesign in business information processes[J]. Collaboration Systems and Technology Track, 1998, 1(1): 240-252

[9] 孙毅, 张莉. 企业过程分析与监控技术的研究[D]. 北京: 北京航空航天大学, 2005