

PID 神经网络的研究和改进

卢 萍, 金朝永

(广东工业大学 应用数学学院, 广东 广州 510520)

摘要: 针对 PID 神经网络能根据控制效果进行在线自学习调整及具有无静差控制效果等特点, 通过分析 PIDNN 控制算法存在的局限性, 提出了对误差进行分级的改进算法. 对改进的算法进行仿真, 结果证明该方法能有效地改善网络的性能.

关键词: 收敛速度; 误差迭代; 神经网络; 仿真; PID 控制

中图分类号: TP18

文献标志码: A

文章编号: 1007-7162(2011)04-0055-04

人工神经网络是对人类大脑系统的一阶特性的描述, 具有自学习、自组织和自适应性, 可以处理一些环境信息十分复杂、知识背景不清楚和推理不明确的问题. PID 神经网络(PIDNN)是将 PID 控制规律融进神经网络之中, 实现了神经网络和 PID 控制规律的结合. 它是多层前向神经网络, 与一般的多层前向神经网络不同的是: PIDNN 的隐含层是由比例元、积分元、微分元组成, 这些神经元的输入-输出函数分别为比例、积分和微分函数, 既有静态

的, 又有动态的, 具有易适应控制系统的要求. 虽然它有很多的优点, 但它仍然存在一些不足, 针对这些不足, 本文主要对参数学习误差进行分析和改进, 仿真结果表明能提高网络的性能.

1 SPIDNN 的结构形式

PIDNN^[1]的基本形式是 $2 \times 3 \times 1$ 的结构, 如图 1 所示, 包括输入层的两个神经元、隐含层的 3 个神经元、输出层的一个神经元.

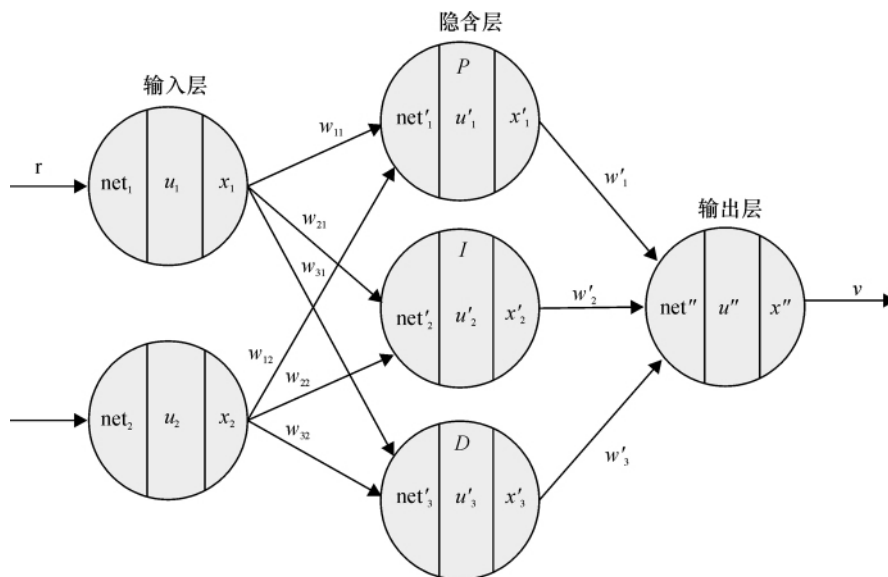


图 1 SPIDNN 的结构图

1.1 SPIDNN 的误差前向算法

1.1.1 输入层

在任意采样时刻 k , 神经元的输入

$$\text{net}_1(k) = r(k), \quad \text{net}_2(k) = y(k). \quad (1)$$

神经元的状态

$$u_i(k) = \text{net}_i(k). \quad (2)$$

神经元的输出

收稿日期: 2011-04-21

作者简介: 卢 萍(1986-), 女, 硕士研究生, 主要研究方向为系统的智能控制和优化.

$$x_i(k) = \begin{cases} 1 & u_i(k) > 1, \\ u_i(k) & -1 \leq u_i(k) \leq 1, \\ -1 & u_i(k) < -1. \end{cases} \quad (3)$$

1.1.2 隐含层

隐含层有3个神经元,分别为比例元、积分元和微分元,各自的输入均值中输入层至隐含层的加权总和,即为

$$\mathbf{net}'_j(k) = \begin{pmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \mathbf{net}'_1(k) \\ \mathbf{net}'_2(k) \\ \mathbf{net}'_3(k) \end{pmatrix}. \quad (4)$$

神经元状态中比例元的状态函数

$$u'_1(k) = \mathbf{net}'_1(k). \quad (5)$$

神经元状态中积分元的状态函数

$$u'_2(k) = u'_2(k-1) + \mathbf{net}'_2(k). \quad (6)$$

神经元状态中微分元的状态函数

$$u'_3(k) = \mathbf{net}'_3(k) - \mathbf{net}'_3(k-1). \quad (7)$$

神经元的输出

$$x'_j(k) = \begin{cases} 1 & u'_j > 1, \\ u'_j & -1 \leq u'_j \leq 1, \\ -1 & u'_j < -1. \end{cases} \quad (8)$$

1.1.3 输出层

神经元的输出

$$\mathbf{net}''(k) = w'_1 x'_1(k) + w'_2 x'_2(k) + w'_3 x'_3(k), \quad (9)$$

状态

$$u'' = \mathbf{net}''(k), \quad (10)$$

输出

$$x''(k) = \begin{cases} 1 & u''(k) > 1, \\ u''(k) & -1 \leq u''(k) \leq 1, \\ -1 & u''(k) < -1. \end{cases} \quad (11)$$

而 SPIDNN 的输出值与输出层神经元的输出值相同,即

$$v(k) = x''(k). \quad (12)$$

1.2 SPIDNN 的误差反向传播算法

若该 SPIDNN 实际输出矢量的集合 $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p, \dots)^T$, 对应的理想输出矢量的集合是 $\mathbf{V}' = (\mathbf{v}'_1, \mathbf{v}'_2, \dots, \mathbf{v}'_p, \dots)^T$, l 为采样点数. SPIDNN 训练和学习的目的是使偏差平方均值 $E = \frac{1}{l} \sum_{k=1}^l [v'(k) - v(k)]^2$ 为最小.

根据梯度法,按附加动量法调节网络权值,设学习步长为 η ,各层权值的迭代方程经过 k 步训练和学习后,各层权值的迭代方程为 $w(k+1) = w(k) -$

$$\eta \frac{\partial J}{\partial w}.$$

1.2.1 隐含层至输出层

$$w'_j(k+1) = w'_j(k) - \eta_j \frac{\partial E}{\partial w'_j}, \quad (13)$$

式中

$$\frac{\partial E}{\partial w'_j} = \frac{\partial E}{\partial v} \frac{\partial v}{\partial x''} \frac{\partial x''}{\partial u''} \frac{\partial u''}{\partial \mathbf{net}''} \frac{\partial \mathbf{net}''}{\partial w'_j} = -\frac{2}{l} \sum_{k=1}^l [v'(k) - v(k)] x'_j(k). \quad (14)$$

即得到 SPIDNN 隐含层至输出层的连接权值的算法修改为

$$w'_j(k+1) = w'_j(k) + \frac{\eta_j}{l} \sum_{k=1}^l [v'(k) - v(k)] x'_j(k) \quad j = 1, 2, 3.$$

1.2.2 输入层至隐含层

$$w_{ij}(k+1) = w_{ij}(k) - \eta_{ij} \frac{\partial E}{\partial w_{ij}}, \quad (15)$$

则输入层至隐含层的权值变化为

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial v} \frac{\partial v}{\partial x''} \frac{\partial x''}{\partial u''} \frac{\partial u''}{\partial \mathbf{net}''} \frac{\partial \mathbf{net}''}{\partial x'_j} \frac{\partial x'_j}{\partial u'_j} \frac{\partial u'_j}{\partial \mathbf{net}'_j} \frac{\partial \mathbf{net}'_j}{\partial w_{ij}}. \quad (16)$$

对于因式 $\frac{\partial u'_j}{\partial \mathbf{net}'_j}$, 由于被控对象未知,不能直接求出来,所以根据求导的定义有

$$\frac{\partial u'_j}{\partial \mathbf{net}'_j} \approx \frac{\Delta u'_j}{\Delta \mathbf{net}'_j} = \frac{u'_j(k) - u'_j(k-1)}{\mathbf{net}'_j(k) - \mathbf{net}'_j(k-1)}. \quad (17)$$

当 $\mathbf{net}'_j(k) - \mathbf{net}'_j(k-1) \approx 0$ 时,式(17)的结果可能会趋于无穷大,无法进行下面的计算,需要采用近似的方法.式(17)绝对值的大小影响权重值的收敛变化速度,它的正负影响权重值的收敛方向.相对于收敛速度,收敛方向更为重要,所以用 $\mathbf{net}'_j$ 和 u'_j 的相对变化量的符号函数近似代替式(17),即

$$\frac{\Delta u'_j}{\Delta \mathbf{net}'_j} = \frac{u'_j(k) - u'_j(k-1)}{\mathbf{net}'_j(k) - \mathbf{net}'_j(k-1)} \approx \frac{u'_j(k) - u'_j(k-1)}{\mathbf{net}''_j(k) - \mathbf{net}''_j(k-1)}, \quad (18)$$

则

$$\frac{\partial E}{\partial w_{ij}} = -\frac{2}{l} \sum_{k=1}^l [v'(k) - v(k)] \times \operatorname{sgn} \frac{u'_j(k) - u'_j(k-1)}{\mathbf{net}'_j(k) - \mathbf{net}'_j(k-1)} w'_{ij} x_i. \quad (19)$$

由以上各式,即得到 SPIDNN 输入层至隐含层的连接权值算法修改为

$$w_{ij}(k+1) = w_{ij}(k) + \frac{2}{l} \sum_{k=1}^l [v'(k) - v(k)] \times \operatorname{sgn} \frac{u'_j(k) - u'_j(k-1)}{\mathbf{net}'_j(k) - \mathbf{net}'_j(k-1)} w'_{ij} x_i. \quad (20)$$

由 SPIDNN 控制系统的结构和算法可知, SPIDNN 控制系统的算法不依赖于被控对象参数,而是根据控制效果在线自主进行学习,调整网络连接权值,实现对被控对象的控制。

2 SPIDNN 的改进

因为 SPIDNN 对无 PID 先验知识的控制系统,初始权值只能取随机数^[2]。这样收敛的方向是任意的,容易引起网络性能不稳定等缺点,而当误差很小时,收敛很慢。基于此,对 SPIDNN 作一个收敛控制学习误差 ε_0 (不宜过小),对其进行分级^[3]。分级的方法为: $\varepsilon_i = \left(\frac{p}{q}\right)^i \varepsilon_0$, 其中 $p < q$, 依次取 $\varepsilon_0, \varepsilon_1, \dots$, ε_N 作为收敛控制参数,让网络对学习集样本进行学习训练,其控制算法^[4]如下。

(1) 初始化系统参数 p 和 q , 给定学习中的误差 ε_0 以及迭代步数 n 及最小参数 $\min \varepsilon$ 。

(2) 通过神经网络模拟,得到当前误差 $\varepsilon_i, I = I + 1$ 。

(3) 如果 $\varepsilon_i < \min \varepsilon$, 则跳转执行第(4)步; 如果 I 大于 n , 执行第(4)步; 否则返回第(2)步继续执行。

(4) 保存参数,结束。

分级迭代法是在给定 ε_0 之后,确定 p 和 q ,按公式依次确定 $\varepsilon_0, \varepsilon_1, \dots, \varepsilon_N$ 。然后用 ε_1 代替 $\varepsilon_0, \varepsilon_2$ 代替 ε_1 。按常规的方法和思路进行训练和迭代,对当 $\varepsilon_i = \varepsilon_0$ 时, ε_0 相对较大,在第一级误差的迭代过程中,对样本的误差要求不高,不需要过分地迁就。此时收敛得比较快,且此时的权值 w 对 ε_0 也有所适应,然后再用修正好的在 ε_1 中迭代。因此, ε_1 只为 ε_0 的 $\frac{q}{p}$, 小于 ε_0 。因此迭代得比较平稳,而迭代也会很快收敛,依次进行下去,直至第 n 级误差迭代收敛为止。这种迭代实质上是一种循序渐进思路的实现^[5-6]。

3 仿真结果

对于二阶环节,设其传递函数为^[7]

$$G(s) = \frac{Y(s)}{V(s)} = \frac{1 - e^{-\tau s}}{s} \frac{K_m}{s(1 + T_m s)},$$

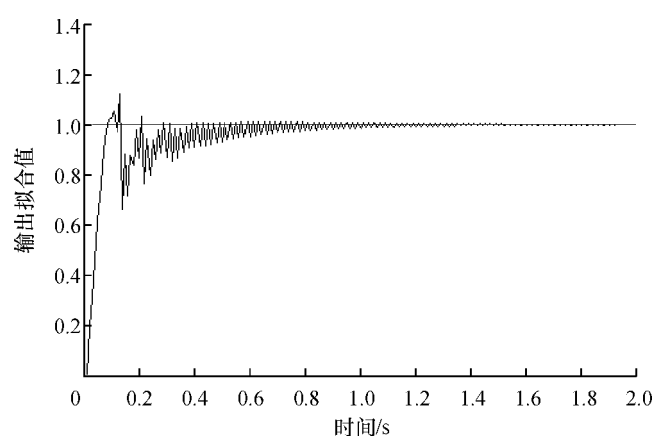
式中: $Y(s)$ 为对象输入, $V(s)$ 为对象输出, K_m 为对象放大系数, T_m 为对象时间常数, T 为系统采样周期。取 $K_m = 10, T = T_m = 0.025$, 则可以得到以下的二阶离散化算式:

$$y(k+1) = 1.368 y(k) - 0.368 y(k-1) + 0.092 v(k) + 0.066 v(k-1).$$

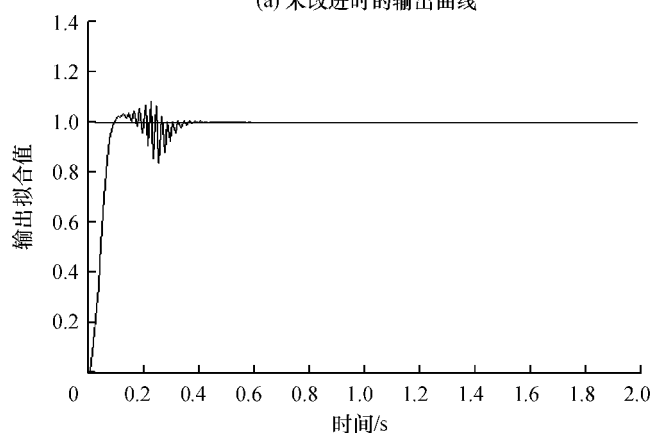
系统给定的阶跃函数 $r(k) = 1$, 初始步长 $\eta = 0.05$, 动量因子 $\vartheta = 0.2$, 初始权值选区间上的随机数, 采样周期 $t_s = 0.001$ s, 取输入层至隐含层的初始权值都为 1, 隐含层至输出层的初始权值为 0.1, 用 Matlab^[8-10] 进行程序仿真, 结果如图 2 所示。

图 2 为 SPIDNN 控制系统在无先验知识时的仿真结果, 其中图 2(a) 是随机选取权值初值时的系统输出曲线, 图 2(b) 是随机选取权值初值时改进后的系统的误差曲线。

由仿真图可以看出, 改进后的系统在同等条件下, 能在短时间内经过网络的自学习、自调整达到比较好的动静态性能。



(a) 未改进时的输出曲线



(b) 改进后的输出曲线

图 2 仿真结果

4 结论

本文通过对误差进行分级迭代,使系统先在一个大范围内适应,接着在小范围适应,通过这个循序渐进的过程,使得系统的稳定性能得到一定的改善。同时将改进后的 SPIDNN 对二阶时延对象进行仿真。仿真表明,改进后的 SPIDNN 控制系统得到了较好的收敛性。

参考文献:

- [1] 舒怀林. PID 神经网络及其控制系统 [M]. 北京: 国防工业出版社, 2006.
- [2] 李文峰, 刘军营, 盛蕾. 一种改进的基于神经网络模式的 PID 控制研究 [J]. 系统仿真技术, 2007, 3(4): 217-220.
- [3] 孙立雷, 朱玮, 王春阳. 基于改进 BP 神经网络的优化 PID 控制器应用研究 [J]. 仪器仪表学报, 2006, 27(12): 388-390.
- [4] 蒋宗礼. 人工神经网络导论 [M]. 北京: 高等教育出版社, 2005.
- [5] 胡伍生. 神经网络理论及其工程应用 [M]. 北京: 测绘出版社, 2006.
- [6] 毕长春, 李柠, 黄道. 蓄热式加热炉钢温预报与炉温优化设定研究 [J]. 自动化学报, 2004, 30(3): 476-480.
- [7] 翟尧杰. 嵌入式 PID 神经网络调节器的研究 [D]. 广州大学机械与电气工程学院, 2010.
- [8] 刘金琨. 先进 PID 控制 MATLAB 仿真 [M]. 2 版. 北京: 电子工业出版社, 2006.
- [9] 宋滨, 谷丽娜. 基于神经网络的 PID 控制 [J]. 青岛大学学报, 2001, 6(3): 90-91.
- [10] 王洪东, 李丽, 白金泉. MATLAB 在控制工程中的应用 [J]. 控制工程, 2003, 10(3): 273-275.

Research and Improvement of PID Neural Network

Lu Ping, Jin Chao-yong

(Faculty of Applied Mathematics, Guangdong University of Technology, Guangzhou 510520, China)

Abstract: Considering that the neural network of Proportion Integration Differentiation (PID) can adjust online self-learning and has no static errors, it proposed an improved algorithm for classification of errors by analyzing the defects of the control method. Simulation was done for this improved algorithm. The results indicate that the improved algorithm can improve the performance of the network effectively.

Key words: convergent velocity; iteration error; neural network; simulation; proportion integration differentiation (PID)

(上接第47页)

- [3] Kulasinghe P, Bettayeb S. Embedding binary trees into crossed cubes [J]. IEEE Transactions on Computers, 1995, 44(7): 923-929.
- [4] Yang M C, Li T K. Fault-tolerant cycle-embedding of crossed cubes [J]. Information Processing Letters, 2003, 88(4): 149-154.
- [5] Hung H, Fu J, Chen G. Fault-free Hamiltonian cycles in crossed cubes with conditional link faults [J]. Information Science, 2007, 177(24): 5664-5674.
- [6] Dong Q, Yang X. Embedding a long fault-free cycle in a crossed cube with more faulty nodes [J]. Information Processing Letters, 2010, 110(11): 464-468.
- [7] Baolei C, Fan J, Jiwen Y, et al. An algorithm to construct independent spanning trees on crossed cubes [EB/OL]. [2011-06-20]. <http://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=An+algorithm+to+construct+independent+spanning+trees+on+crossed+cubes&x=39&y=17>.
- [8] Efe K. The crossed cube architecture for parallel computation [J]. IEEE Transactions on Computers, Parallel and Distributed Systems, 1992, 3(5): 513-524.
- [9] Efe K. Topological properties of the crossed cube architecture [J]. Parallel Computing, 1994, 20(12): 1763-1775.

A Kind of Automorphism of the Crossed Cube

Li Lin, Guo Da-chang, Ying Chao-jie

(Faculty of Applied Mathematics, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: It determined a kind of automorphism of the crossed cube, and proved that the automorphism formed a group. Moreover, this automorphism group was used to classify crossed cube nodes.

Key words: crossed cube; hypercube; interconnected networks; similar vertices; dimension preserving automorphism