文章编号: 1001-0920(2018)06-1055-07 DOI: 10.13195/j.kzyjc.2017.0060

基于PSO算法的动态模块化神经网络结构设计

卢 超†, 杨翠丽, 乔俊飞

(1. 北京工业大学 信息学部，北京 100124；2. 计算智能与智能系统北京市重点实验室，北京 100124)

摘 要: 针对模块化神经网络结构设计过程中子网络输出不能最优集成的问题,提出一种基于粒子群算法的动态模块化神经网络. 首先,该网络采用数据密度辨识样本分布空间,并更新数据中心;然后,根据输入数据激活相应的子网络,利用PSO算法寻找子网络的最优网络贡献度,并依据贡献度计算子网络的输出权值;最后优化模块化神经网络的集成输出. 通过对非线性函数和时变系统的逼近实验,验证了集成网络中子网络数目可以根据任务动态调整,网络输出的集成权值能够通过PSO算法寻找到最优值,并且训练精度和自适应能力较其他算法均有一定的提高.

关键词: 模块化神经网络；粒子群算法；动态集成；时变系统

中图分类号: TP183 **文献标志码:** A

Dynamic modular neural network structure design based on PSO algorithm

LU Chao†, YANG Cui-li, QIAO Jun-fei

(1. Faculty of Information Technology，Beijing University of Technology，Beijing 100124，China；2. Beijing Key Laboratory of Computational Intelligence and Intelligent System，Beijing 100124，China)

Abstract: In order to solve the problem the sub-network output can not be optimally integrated in a modular neural network(MNN), this paper proposeds a dynamic MNN based on the particle swarm optimization(PSO) algorithm. Firstly, the distribution of samples can be identified and the center of datas can be updated by computing the data density. Secondly, the corresponding sub-networks are activated according to the input datas, then the output weights are calculated by the best contribution degrees which are computed via the PSO algorithm. Finally, a dynamic neural network is completed to optimize the integrated output of the MNN. Based on the approximating experiments of the non-linear function and time-series prediction, it is proved that the number of sub-networks can be adjusted dynamically, and the integrated weights of the neural network can be optimized by using the PSO algorithm. Comparisons with other algorithms demonstrate that the proposed method is more effective in terms of the accuracy and adaptive ability.

Keywords: modular neural network；PSO algorithm；dynamic integrated；time-varying system

0 引 言

模块化神经网络(MNN)采用“集思广益、分而治之”的思想,将复杂的实际问题分解为若干个相互独立但又相互联系的子问题,对于每一个子问题建立相应的神经网络模型,最终解决实际问题[1-2]. 模块化神经网络能够有效地降低神经网络的复杂程度,解决单一神经网络在面对复杂问题时存在的结构设计困难、学习速度慢、学习精度低等问题,已在模式识别、建模优化、智能预测等领域得到了广泛的应用[1-4].

自组织模块化神经网络在传统模块化网络结构的基础上增加了自组织调整机制,可以自适应地调整网络结构,提高模块化网络处理复杂问题的能力,因此得到了广泛的应用. Ramamurti等[5]提出了SGMN(Self-regulating growing multi-experts network)网络,该网络可以随着时变系统的变化自适应调整网络结构. Molina等[6]提出了适用于时变系统的模块化神经网络(SAMNN),该网络可以随着输入信息的变化进行调整. 然而,SAMNN子网络的增加或删除采取固定步长策略,不能及时反映时变系统的时变特性,同时,SAMNN和SGMN中子网络增加或删

收稿日期: 2017-01-15；**修回日期:** 2017-07-17.

基金项目: 国家自然科学基金重点项目(61533002)；国家自然科学基金青年基金项目(61603012).

责任编委: 曹进德.

作者简介: 卢超(1988−), 男, 博士生, 从事智能系统建模优化的研究；乔俊飞(1968−), 男, 教授, 博士生导师, 从事智能特征建模、智能优化控制、智能信息处理等研究.

†通讯作者. E-mail: luchaobs@emails.bjut.edu.cn

除的条件由子网络输出的局部误差决定,使得网络的增长速度缓慢,而且子网络的结构在整个学习过程中都固定不变,降低了对时变系统的自适应能力. Qiao等[7]提出了基于动态模块化神经网络结构设计方法,依据减法聚类对输入信息进行分类,动态调整子网络的结构和参数,提高了模块化神经网络的精度和适应性. 自组织模块化神经网络在结构设计过程中具有很多优势[5-7],但没有考虑子网络输出权值最优集成的问题,通常集成权值被赋予0～1之间的固定值或通过距离测度等方法获得.

为了解决上述问题,Cimino等[8]采用每个子网络对本网络和相邻子网络样本同时进行训练的方法,并在子网络整合中进行集成,对于任一给定输入,与该输入相邻的所有子网络均参与信息的处理. 但上述方法中,MNN的每个子网络都对所有相邻区域的样本进行训练,这势必会增大子网络的规模. Zhou等[9]对集成学习问题进行研究,提出了“Many could be better than all”的思想,并证明从与输入相邻的所有子网络中选择部分子网络进行集成可以获得更好的性能,同时提出了一种基于遗传算法的权重优化算法[10],使得网络更具有灵活性. 但网络在调整过程中,如果K值选择不合理,则可能降低网络性能. Tseng等[11]等提出了一种分层模糊模块化神经网络用以预测飞机航行轨迹问题,该方法采用遗传算法集合子网络,但网络整合过程中遗传算法计算量较大,增加了计算复杂度. 综上所述,对于一个时变的输入信息,如何自适应地选择网络结构并选取表征子网络输出贡献度的最优权值,仍然是自组织模块化神经网络研究的难题.

本文提出一种基于PSO算法的动态模块化神经网络(PSO-DMNN). PSO-DMNN方法依据输入数据,激活相应的子网络,通过PSO算法寻找最优的网络贡献度,并按照贡献度计算子网络的输出权值,完成网络结构的自组织和参数自调整. 最后通过仿真实验验证了PSO-DMNN的有效性.

1 PSO-DMNN网络结构设计

1.1 模块化神经网络结构

模块化神经网络的结构如图1所示,共分为5层:输入层、任务分解层、任务分配层、子网络层、集成输出层.

第1层:输入层. 该层有n个输入节点,n为k时刻输入向量$x(k)$的维数,$\boldsymbol{x}(k)=[x_1(k),\cdots,x_n(k)]^{\mathrm{T}}$;

第2层:任务分解层. 该层包括一个RBF神经元,第h个RBF神经元的激活函数为

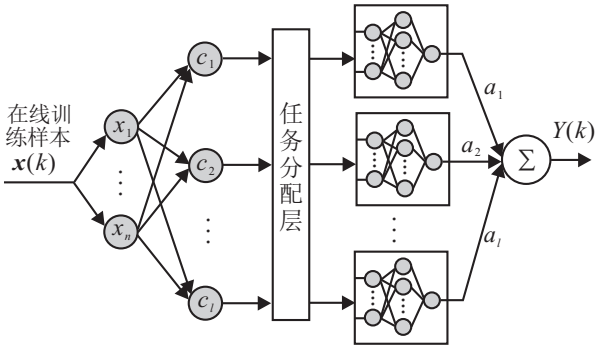

图1 模块化神经网络结构框图

$$\kappa_h(\boldsymbol{x}(k),\boldsymbol{c}_h)=\mathrm{e}^{(\|\boldsymbol{x}(k)-\boldsymbol{c}_h\|^2/\delta_h^2)}. \tag{1}$$

其中:$\boldsymbol{c}_h$为第h个RBF神经元的数据中心,δ_h为扩展宽度. 在模块化神经网络中,$\kappa_h(\boldsymbol{x}(k),\boldsymbol{c}_h)$被定义为硬极限函数,即

$$\kappa_h(\boldsymbol{x}(k),\boldsymbol{c}_h)=\begin{cases}1,\ \|\boldsymbol{x}(k)-\boldsymbol{c}_h\|\leqslant\delta_h;\\ 0,\ \|\boldsymbol{x}(k)-\boldsymbol{c}_h\|>\delta_h.\end{cases} \tag{2}$$

模块化神经网络的物理意义是,在输入样本空间中,以$\boldsymbol{c}_h$为球心,以δ_h为半径的超球面. 当输入样本在超球面以外时输出为0,在超球面以内时输出为1,实现对输入样本的分解.

第3层:任务分配层. 该层有l个激活函数,根据$\kappa_h(\boldsymbol{x}(k),\boldsymbol{c}_h)$的输出,若$\kappa_h(\boldsymbol{x}(k),\boldsymbol{c}_h)$输出为1,则激活对应的子网络,对输入样本进行学习;若输出为0,则不激活子网络. 在模块化神经网络中,当k时刻的输入样本$x(k)$落在多个子网络的重叠区域时,该样本会激活多个子网络参与学习,从而提高网络学习的精度.

第4层:子网络层. 该层包括l个子网络,每个子网络均为前馈神经网络,通过学习分配来的样本,更好地学习样本的信息.

设k时刻模块化神经网络中第h个子网络的网络结构为n-m-1(n个输入节点,m个隐节点,1个输出节点),该子网络的输出为

$$y_h(k)=\sum_{i'=1}^{m}w_{i'}(k)f\Big(\sum_{i=1}^{n}v_{ii'}(k)x_i(k)\Big). \tag{3}$$

其中:$w_{i'}$为子网络隐含层节点与输出节点之间的连接权值,$v_{ii'}$为子网络中输入节点与隐节点之间连接权值,$f(x)=1/(1+\mathrm{e}^{-x})$为隐节点激活函数.

第5层:集成输出层. 该层的主要功能是对k时刻参与学习的子网络的输出进行集成,集成输出为

$$Y(k)=\sum_{g'=1}^{R}\alpha_{g'}(k)y_{g'}(k). \tag{4}$$

其中:R为k时刻参与学习的子网络个数,满足$1\leqslant$

$R \leqslant l$; $\alpha_{g'}(k)$为参与学习的子网络集成权值; $y_{g'}(k)$为k时刻参与学习的第g'个子网络的输出.

1.2 数据密度计算

数据密度能够反映出时变系统输入数据的聚集程度. 本文通过减法聚类数据密度的方法调整网络中心和结构[7],其具体方法如下:

在k时刻,如果已经存在l个聚类中心$\boldsymbol{c}_h(h = 1,2,\cdots,l)$,则需计算当前数据与原始聚类中心的密度并进行比较. $\boldsymbol{x}(k)$密度计算公式为

$$P_k(\boldsymbol{x}(k)) = \frac{k-1}{(k-1)(\gamma(k)+1) - 2\boldsymbol{\eta}(k)\boldsymbol{x}(k) + \sigma(k)}, \tag{5}$$

$$\boldsymbol{\eta}(k) = \sum_{j=1}^{k-1} \boldsymbol{x}^{\mathrm{T}}(j), \tag{6}$$

$$\sigma(k) = \sum_{j=1}^{k-1} \boldsymbol{x}^{\mathrm{T}}(j)\boldsymbol{x}^{\mathrm{T}}(j), \tag{7}$$

$$\gamma(k) = \boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{x}(k). \tag{8}$$

参数递推公式如下:

$$\boldsymbol{\eta}(k) = \boldsymbol{\eta}(k-1) + \boldsymbol{x}^{\mathrm{T}}(k-1), \tag{9}$$

$$\sigma(k) = \sigma(k-1) + \boldsymbol{x}^{\mathrm{T}}(k-1)\boldsymbol{x}(k-1). \tag{10}$$

对于聚类中心$\boldsymbol{c}_h$, k时刻的聚类中心密度为

$$P_k(\boldsymbol{c}_h) = \frac{(k-1)P_{k-1}(\boldsymbol{c}_h)}{(k-2) + P_{k-1}(\boldsymbol{c}_h) + \zeta(k)P_{k-1}(\boldsymbol{c}_h)}, \tag{11}$$

其中$\zeta(k) = (\boldsymbol{c}_h - \boldsymbol{x}(k))^{\mathrm{T}}(\boldsymbol{c}_h - \boldsymbol{x}(k))$可由当前数据计算$\boldsymbol{x}(k)$得到.

1.3 聚类中心的动态调整

模块化神经网络需要对输入数据不断学习,数据中心根据以下3种情况进行动态调整.

1) 聚类中心增加.

$$\text{If } \min_{h=1,\cdots,l} \|\boldsymbol{x}(k) - \boldsymbol{c}_h\|_2 > r_1 \text{ and } P_k(\boldsymbol{x}(k)) > \bar{\epsilon},$$

$$\text{Then } \boldsymbol{c}_{l+1} = \boldsymbol{x}_k,\ l = l+1.$$

其中: r_1和$\bar{\epsilon}$为预先设定的阈值,即

$$r_1 = (0.1 \sim 0.5) \times \max_{j' \neq j'', j', j'' = 1,\cdots,N} \|\boldsymbol{x}(j') - \boldsymbol{x}(j'')\|_2,$$

$$\bar{\epsilon} = 0.5 \max_{h=1,\cdots,l} P_{k-1}(\boldsymbol{c}_h),$$

N为样本个数.

情况1)表明,新数据与最近的聚类中心的距离足够大,而且其密度也大于给定的阈值,数据比周围其他数据的描述能力更强,意味着一个新聚类中心的出现,因此对应增加了一个新的子网络.

2) 聚类中心转移.

$$\text{If } \min_{h=1,\cdots,l} \|\boldsymbol{x}(k) - \boldsymbol{c}_h\|_2 \leqslant r_1 \text{ and } P_k(\boldsymbol{x}(k)) > P_k(\boldsymbol{c}_h),$$

$$\text{Then } \boldsymbol{c}_h = \boldsymbol{x}_k,$$

其中$t = \arg \min_{h=1,\cdots,l} \|\boldsymbol{x}(k) - \boldsymbol{c}_h\|_2$.

情况2)表明,新数据与聚类中心距离很近,而且该新数据的密度大于该聚类中心的密度值,说明聚类中心发生了迁移,需要调整原有的聚类中心.

3) 聚类中心合并.

$$\text{If } \min_{h \neq h',\, h, h' = 1,\cdots,l} \|\boldsymbol{c}_h - \boldsymbol{c}_{h'}\|_2 \leqslant r_2,$$

$$\text{Then } \boldsymbol{c}_h = \boldsymbol{c}_{h_{\mathrm{new}}},\ l = l-1.$$

其中:若$P_k(\boldsymbol{c}_h) > P_k(\boldsymbol{c}_{h'})$,则$h_{\mathrm{new}} = h$,否则$h_{\mathrm{new}} = h'$; $r_2 = (0.5 \sim 0.7)r_1$.

情况3)表明,当两个聚类中心距离很近时,可以通过判别数据密度合并聚类中心.

1.4 基于PSO算法的网络集成方法

本文在进行模块化神经网络动态集成时,让样本激活的子网络参与学习,通过PSO算法确定一组最优的输出贡献度,并依据网络贡献度计算子网络的输出权值,从而优化网络结构,提高精度.

粒子群算法(PSO)是一种基于种群的随机搜索算法,其通过模拟鸟群飞行过程中的协作行为,设计粒子间的相互协作及信息共享机制,并记忆自身及群体的历史运动信息,协助粒子当前的运动行为,从而更好地在复杂空间中寻优. 这种算法被用来解决多种优化问题,已在神经网络、模式识别等领域取得了广泛应用.

PSO算法基本概念如下:

PSO算法中每个粒子都被视为D维可行解空间中的一个解,粒子位置可表示为

$$\boldsymbol{a}_q = [a_{q,1}, a_{q,2}, \cdots, a_{q,D}]. \tag{12}$$

其中: D是搜索空间维数; $q = 1,2,\cdots,s$, s是粒子数. 粒子速度可以表示为

$$\boldsymbol{v}_q = [v_{q,1}, v_{q,2}, \cdots, v_{q,D}]. \tag{13}$$

在搜索过程中,算法通过适应度函数值(Fitness value)来评价粒子的"好坏"程度,每个粒子搜索到的最好位置称为个体极值$\boldsymbol{p}_q(t)$,有

$$\boldsymbol{p}_q(t) = [p_{q,1}(t), p_{q,2}(t), \cdots, p_{q,D}(t)]. \tag{14}$$

而整个群体搜索到的最好位置称为全局极值,记录群体全局最优解为

$$\boldsymbol{g}(t) = [g_1(t), g_2(t), \cdots, g_D(t)]. \tag{15}$$

每个粒子根据自身的速度、个体极值和全局极值更新其速度和位置. 速度及位置更新公式如下:

$$v_{q,d}(t+1)=v_{q,d}(t)+c_1'r_1'(p_{q,d}(t)-a_{q,d}(t))+ c_2'r_2'(g_d(t)-a_{q,d}(t)), \tag{16}$$

$$a_{q,d}(t+1)=a_{q,d}(t)+v_{q,d}(t+1). \tag{17}$$

其中: c_1' 和 c_2' 是加速度常数,且一般在 $[0,2]$ 之间取值; r_1' 和 r_2' 是 $[0,1]$ 之间均匀分布的随机数. 通过粒子适应度值更新粒子个体最优,即

$$\boldsymbol{p}_q(t+1)=\begin{cases}\boldsymbol{p}_q(t),\ f(\boldsymbol{a}_q(t+1))\geqslant f(\boldsymbol{p}_q(t));\\ \boldsymbol{a}_q(t+1),\ \text{otherwise}.\end{cases} \tag{18}$$

其中函数 $f(a_q(t))$ 是适应度函数,用于表示解的优劣. 通过选择个体最优、适应度最小的解作为全局最优解,有

$$\boldsymbol{g}(t+1)=\arg\min_{\boldsymbol{p}_q}(f(\boldsymbol{p}_q(t+1))),\ 1\leqslant q\leqslant s. \tag{19}$$

由粒子的速度更新公式可知,粒子速度主要受3部分影响:

1) 上一时刻的速度 $v_{q,d}(t)$,即粒子受惯性行为的影响;

2) $c_1'r_1'(p_{q,d}(t)-a_{q,d}(t))$,即粒子的自学习过程;

3) $c_2'r_2'(g_d(t)-a_{q,d}(t))$,即粒子向全局最优靠近过程.

PSO迭代寻优的操作步骤如下:

1) 初始化种群粒子数. 粒子个数通常为20 ∽ 80,本文设置种群粒子数为30,最大迭代次数为50.

2) 初始化粒子的位置及速度信息. 设置初始粒子位置和速度为0～1之间的随机值,加速度常数 $c_1'=c_2'=1.494\,45$.

3) 依据公式 $v_{q,d}(t+1)=v_{q,d}(t)+c_1'r_1'(p_{q,d}(t)-a_{q,d}(t))+c_2'r_2'(g_d(t)-a_{q,d}(t))$ 更新粒子速度信息,通过公式 $a_{q,d}(t+1)=a_{q,d}(t)+v_{q,d}(t+1)$ 更新位置信息.

4) 由粒子适应度函数计算每个粒子适应度值,依据式(14)和(15)选择个体最优 $\boldsymbol{p}_q(t)$ 及全局最优 $\boldsymbol{g}(t)$. 判断是否满足终止条件,若满足则转到5),否则转到3).

5) 将粒子最优解作为一组最优参数输出,即为优化结果.

为了获得最优的权值,文中采用实际输出与网络输出之间的相对误差函数(RMSE)作为PSO算法的适应度函数,用 k 取代PSO算法中的参数 t,得到适应度函数公式如下:

$$f(\boldsymbol{\beta}(k))=\sqrt{\frac{1}{N}\sum_{k=1}^{N}(Y_\beta(k)-y(k))^2}. \tag{20}$$

其中: $\boldsymbol{\beta}(k)=[\beta_1(k),\cdots,\beta_{g'}(k),\cdots,\beta_R(k)]$ 为子网络输出贡献度; $y(k)$ 为实际输出; N 为样本数; $Y_\beta(k)$ 为依据贡献度计算的网络输出,即

$$Y_\beta(k)=\sum_{g'=1}^{R}\beta_{g'}(k)y_{g'}(k). \tag{21}$$

子网络的输出权值为

$$\alpha_{g'}(k)=\frac{\beta_{g'}(k)}{\beta_1(k)+\beta_2(k)+\cdots+\beta_R(k)}. \tag{22}$$

通过对输入数据的训练,当适应度函数最小时,计算得到最优权值. 同时,记录参与学习的子网络数目、最优集成权值和输入数据,建立模块化神经网络子网络参数历史库. 在预测过程中,设定范围阈值,若预测输入数据与历史库中数据的欧氏距离小于设定的阈值 δ,则激活相应的子网络和集成权值,通过学习获得模块化神经网络的预测值.

2 PSO-DMNN结构设计及学习算法

PSO-DMNN网络结构调整学习方法如下.

Step 1: 采集 k 时刻的学习样本 $(\boldsymbol{x}(k),\boldsymbol{y}(k))$,在线递推计算 $P_k(\boldsymbol{x}(k))$ 和 $P_k(\boldsymbol{c}_h)$.

Step 2: 通过输入数据,依据3种不同情况在线更新数据中心 $\boldsymbol{c}_h$ 的值,同时动态调整RBF神经元对应的隐含层子网络.

Step 3: 计算 $\boldsymbol{x}(k)$ 到各激活神经元数据中心的值,计算欧氏距离从而判断是否激活该子网络.

Step 4: 选择同时被激活的子网络对输入数据 $\boldsymbol{x}(k)$ 学习,并通过PSO算法计算最合适的输出权值,对各子网络进行输出集成.

Step 5: 若集成输出没有达到精度要求,则:

1) 计算第 h 个子网络独立学习时的输出误差 $E_h(k)$ 和误差下降趋势 $\text{PE}_h(k)$,其中 $\text{PE}_h(k)=(E_h(k)-E_h(k-1))/E_h(k)\times 100\,\%$;

2) 若 $E_h(k)$ 小于精度要求且子网络累计激活次数 $>\chi$,其中 χ 为设定激活次数,则跳转4),否则转3);

3) 若 $\text{PE}_h(k)>\psi$,其中 ψ 为设定阈值,则转Step 6,否则表明学习有可能陷入局部极小点,此时应跳出极小点,于是 $k+1$ 时刻子网络subNet h 输入层与隐层间的权连接矩阵为

$$\boldsymbol{W}_{h_{\text{sub}}}(k+1)=-\boldsymbol{W}_{h_{\text{sub}}}(k),$$

转至Step 6;

4) 在subNet h 隐层增加一个隐节点,即选择subNet h 中输出贡献最大的隐节点 h_1,将其分裂为两

个隐节点h_2和h_3,新隐节点h_2和h_3的连接权为$w_2=(1+\xi)w_1, w_3=-\xi w_1$,其中$\xi$为很小的随机数.

Step 6: 达到精度要求或满足条件时结束,否则$k=k+1$,返回Step 1.

3　仿真实验

本文通过两个实验来验证PSO-DMNN方法的有效性. 实验1为非线性动态系统辨识,实验2为分段阶跃的非线性系统. 评价函数采用均方根误差(Root mean square error)和神经元个数作为性能评价指标,RMSE的计算公式如下:

$$\mathrm{RMSE}=\sqrt{\frac{1}{N}\sum_{k=1}^{N}(Y(k)-y(k))}. \tag{23}$$

其中: $y(k)$为实际输出,$Y(k)$为网络输出,N为样本数. RMSE值越小,训练测试的精度越高.

本文采用的PSO算法初始参数为: 种群粒子数30,加速度常数$c_1'=c_2'=1.494\,45$,最大迭代次数50次; 粒子位置和速度为0～1之间的随机值. PSO-DMNN网络子网络中神经元个数为4,初始子网络数目为1个,其子网络动态调整过程中初始值$r_2=0.5r_1, r_1=(0.1\backsim 0.5)\times\max\limits_{j'\neq j'',j',j''=1,\cdots,N}\|\boldsymbol{x}(j')-\boldsymbol{x}(j'')\|_2$,调整过程中激活次数$\chi=5$,设定值阈值$\psi=0.2, \delta=0.02$.

3.1　实验1

选取动态非线性系统能够预测的一个基准问题作为仿真对象,该基准问题常用来检测神经网络性能,具体描述如下:

$$y(t+1)=\frac{y(t)y(t-1)[y(t)-0.5]}{1+y^2(t)+y^2(t-1)}+u(t). \tag{24}$$

式(24)产生5 500个时间离散的数据样本,其中前5 000($1\leqslant t\leqslant 5\,000$)个用来训练PSO-DMNN网络,后500个用来测试网络的预测效果($5\,000\leqslant t\leqslant 5\,500$). $u(t)$在区间$[-1.5, 1.5]$之间随机产生. 设置输入为$[y(t); y(t-1); u(t)]$,输出为$[y(t+1)]$进行仿真实验.

图2为实际输出与预测输出的对比,其中实线为实际输出,虚线为网络的预测输出. 由图2可知,网络输出值与实际值误差较小,表明PSO-DMNN在预测跟踪随机波动剧烈的动力学系统时,可以准确预测系统的输出值.

图3为训练RMSE和子网络的结构调整图,在训练初期,网络训练的RMSE随着子网络的调整快速下降,表明网络结构随着输入信息的改变进行调整,同时提高了网络预测精度. 在网络学习的中后期,子网络调整减慢并趋于稳定,预测输出达到精度要求,但由于非线性系统输入$u(t)$为$[-1.5, 1.5]$的随机值,子网络结构在训练后期会有微小波动.

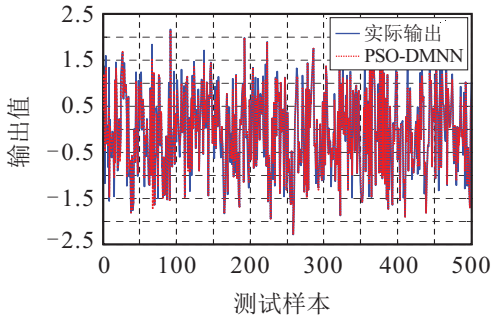

图 2　实际输出与预测输出对比(实验1)

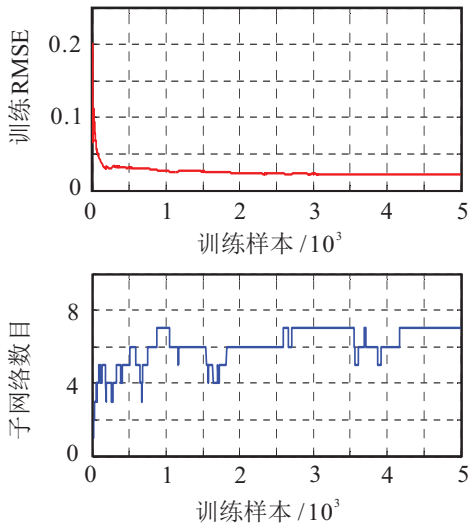

图 3　训练RMSE和子网络结构调整(实验1)

通过20次仿真实验求取训练和测试RMSE的平均值并进行对比. 设定网络的初始神经元个数为1,训练精度为0.01,将PSO-DMNN网络训练预测效果与OSAMNN[7]、MRAN[12]和RANEKF[12]神经网络训练预测效果进行对比(见表1). 显然,本文所提出的神经网络的神经元个数大于MRAN,但小于OSAMNN和RANEKF,训练RMSE和测试RMSE均较小. 在训练时间上,本文方法因增加了PSO 优化算法,所以时间较长.

表 1　不同神经网络模型性能比较(实验1)

网络类型	神经元数	训练RMSE	测试RMSE	训练时间
OSAMNN	40	0.032 9	0.021 1	61.4
MRAN	25	0.036 4	0.028 3	68.8
RANEKF	36	0.028 3	0.027 7	77.7
PSO-DMNN	28	0.022 9	0.020 5	98.4

综上所述,尽管本文方法训练时间较长,但获得了较高精度的训练RMSE、测试RMSE和较为精简的网络结构.

3.2 实验 2

实验 2 为非线性动力学系统辨识. 在该实验中, 神经网络被应用于如下非线性系统辨识:

$$y(t+1)=0.72y(t)+0.025y(t-1)u(t-1)+0.01u^2(t-2)+0.2u(t-3). \tag{25}$$

其中: 非线性系统通过两个输入 $y(t)$ 和 $u(t)$ 来确定输出 $y(t+1)$. 训练输入分为两部分: 一半均匀分布在 $[-2,2]$ 之间, 而另一半输入由 sinusoid 函数 $1.05\times\sin(t/45)$ 确定. 测试输入 $u(t)$ 是一个分段函数, 即

$$u(t)=\begin{cases}\sin(\pi t/25),\ 0<t<250;\\ 1.0,\ 250\leqslant t<500;\\ -1.0,\ 500\leqslant t<750;\\ 0.3\sin(\pi t/25)+0.1\sin(\pi t/32)+\\ 0.6\sin(\pi t/10),\ 750\leqslant t<1\,000.\end{cases} \tag{26}$$

在本次实验中设置训练样本 2 000 组, 测试样本 1 000 组, 做 20 次仿真实验求取训练和测试 RMSE 的平均值, 从而验证网络性能.

图 4 为实际输出与预测输出的对比, 其中实线为实际输出值, 虚线为 PSO-DMNN 模块化神经网络的预测输出值. 图 4 中预测输出值与实际输出值误差较小, 能够准确预测具有分段跳变性质的非线性动力系统, 表明本文提出的 PSO-DMNN 网络通过激活网络的不同子模块, 集成输出达到了很好的预测效果.

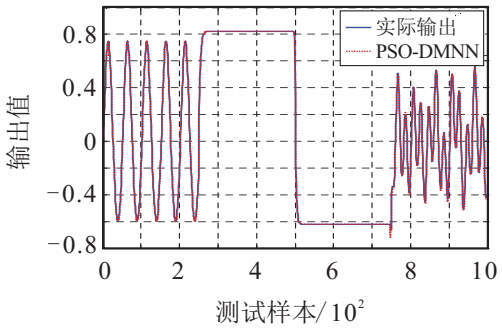

图 4 实际输出与预测输出对比(实验 2)

图 5 为网络训练 RMSE 和子网络结构调整, 从中可以看出, 网络训练误差随时间快速下降, 同时, 子网络结构随着输入信息的改变进行调整, 降低了训练误差, 从而达到了较好的精度要求.

图 6 为实测输出与 PSO-DMNN 网络预测输出的误差值. 本文提出的算法与实际预测值的误差在 0.05 左右; 当 $t=500$ 时, 误差为 0.041, 具有较高的预测精度.

在学习过程中设定网络的初始神经元个数或规则数为 1, 训练精度为 0.01. 将 PSO-DMNN 网络与

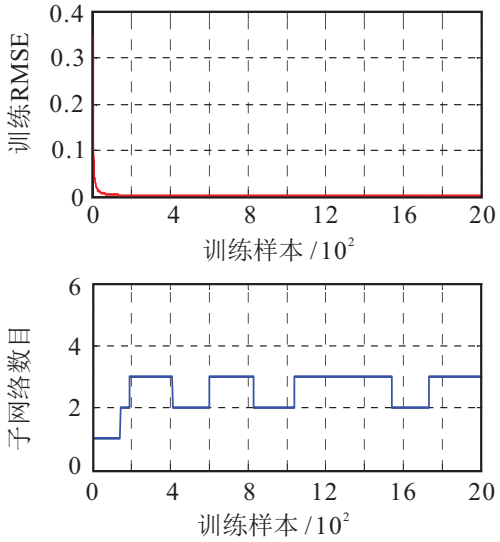

图 5 训练 RMSE 和子网络结构调整(实验 2)

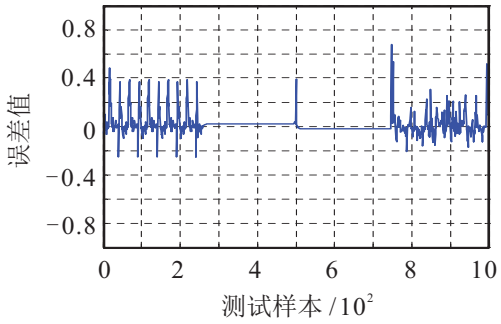

图 6 实际输出与 PSO-DMNN 输出的误差

SRRBF[13]、RSONFN[14] 和 OSAMNN[7] 神经网络进行比较(见表 2). 可见: 本文所提出的网络的训练 RMSE 和测试 RMSE 相较于其他网络为最小, 同时也获得了较为合理的网络结构; 但由于在权值集成过程中采用了 PSO 优化算法, 使得计算量较大, 复杂度较高, 因而增加了训练时间. 因此, 本文提出的算法可以保证较高的精度和较精简的网络结构, 但同时也增加了训练时间.

表 2 不同神经网络模型性能比较(实验 2)

网络类型	神经元数	训练 RMSE	测试 RMSE	训练时间
OSAMNN	16	0.025 0	0.031 7	48.4
SRRBF	14	0.137 1	0.196 0	76.6
RSONFN	9	0.026 0	0.057 1	66.2
PSO-DMNN	12	0.010 7	0.014 6	80.1

4 结 论

针对模块化神经网络不能依据子网络输出完成最优集成的问题, 本文提出了一种基于 PSO 算法的自适应模块化网络结构设计方法. PSO-DMNN 采用数据密度在线辨识分布空间, 激活样本落在重叠区域的子网络, 利用 PSO 算法寻找最优的网络贡献度, 并按照贡献度计算子网络的输出权值, 实现了模块化神

经网络子网络输出的动态集成. 通过与其他神经网络比较可以看出,该网络中子网络数目可以根据任务动态调整,网络输出的集成权值能够通过PSO算法优化寻找到最优值,在有效预测系统输出的同时,提高了网络的训练精度和自适应能力.

在本文的工作中仍有许多值得提高之处,如:在权值优化的过程中,可以采用多目标算法对网络中心、宽度和神经元的个数同时学习,以优化网络结构和参数;在任务分解过程中,可以采用高精度分类方法完成对输入数据类别的划分,从而提高网络性能. 因此,如何最优地实现模块化神经网络任务的分解和集成仍是一个值得研究的问题.

参考文献(References)

[1] Chang Y T, Pantazis D, Leahy R M. To cut or not to cut? Assessing the modular structure of brain networks[J]. NeuroImage, 2014, 91: 99-108.

[2] Sánchez D, Melin P. Optimization of modular granular neural networks using a hierarchical genetic algorithm based on the database complexity applied to human recognition[J]. Information Sciences, 2015, 309: 73-101.

[3] González B, Valdez F, Melin P, et al. Fuzzy logic in the gravitational search algorithm enhanced using fuzzy logic with dynamic alpha parameter value adaptation for the optimization of modular neural networks in echocardiogram recognition[J]. Applied Soft Computing, 2015, 37: 245-254.

[4] Sánchez D, Melin P. Optimization of modular granular neural networks using hierarchical genetic algorithms for human recognition using the ear biometric measure[J]. Engineering Applications of Artificial Intelligence, 2014, 27: 41-56.

[5] Ramamurti V, Ghosh J. Structurally adaptive modular networks for nonstationary environments[J]. IEEE Trans on Neural Networks, 1999, 10(1): 152-160.

[6] Molina V J, Feliu B J, Lopez C J. A modular neural network architecture for step-wise learning of grasping tasks[J]. Neural Networks, 2007, 20(5): 631-645.

[7] Qiao J F, Zhang Z Z, Bo Y C. An online self-adaptive modular neural network for time-varying systems[J]. Neurocomputing, 2014, 125: 7-16.

[8] Cimino M G C A, Pedrycz W, Lazzerini B, et al. Using multilayer perceptrons as receptive fields in the design of neural network[J]. Neurocomputing, 2009, 72(10): 2536-2548.

[9] Zhou Z H, Wu J, Tang W, et al. Combining regression estimators GA-based selective neural network ensemble[J]. Int J of Computational Intelligence and Applications, 2001, 1(4): 341-356.

[10] Zhou Z H, Wu J X, Tang W. Ensembling neural network many could be better than all[J]. Artificial Intelligence, 2002, 137(1): 239-263.

[11] Tseng H C, Almogahed B. Modular neural networks with applications to pattern profiling problems[J]. Neurocomputing, 2009, 72(10): 2093-2100.

[12] Rong H J, Sundararajan N, Huang G B. Sequential adaptive fuzzy inference system(SAFIS) for nonlinear system identification and prediction[J]. Fuzzy Sets Systems, 2006, 157(9): 1260−1275.

[13] Lu C, Han H G, Qiao J F, et al. Design of a self-organizing recurrent RBF neural network based on spiking mechanism[C]. The 35th Chinese Control Conf. Chengdu: IEEE, 2016: 3624-3629.

[14] Lin Y Y, Chang J Y, Lin C T. Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network[J]. IEEE Trans on Neural Networks and Learning Systems, 2013, 24(2): 310-321.

(责任编辑: 李君玲)