DOI:10.13471/j.cnki.acta.snus.2016.04.001

带后续迭代的双极 S 函数激励的 WASD 神经网络*

张雨浓[1,2,3]，肖争利[1,2,3]，丁思彤[1,2,3]，毛明志[1]，刘锦荣[1]
（1. 中山大学数据科学与计算机学院，广东 广州 510006；
2. 华南理工大学自主系统和网络控制教育部重点实验室，广东 广州 510640；
3. 广东顺德中山大学卡内基梅隆大学国际联合研究院，广东 佛山 528300）

摘 要：结合 Levenberg-Marquardt 算法以及权值直接确定法这两种用于神经网络学习训练的方法，提出了一种带后续迭代、面向双极 S（sigmoid）激励函数神经网络的权值与结构确定（weights-and-structure-determination，WASD）方法。该方法与 MATLAB 软件神经网络工具箱相结合，可以解决传统神经网络普遍存在的学习时间长、网络结构难以确定、学习能力和泛化能力有待提高等不足，同时具有较好的可行性和可操作性。以非线性函数的数据拟合为例，计算机数值实验和对比结果证实了 WASD 方法确定出最优隐神经元数和最优权值的优越性，最终得到的 WASD 神经网络具有更为优异的学习性能和泛化性能。

关键词：神经网络；权值与结构直接确定；后续迭代；双极 S 激励函数；数值实验
中图分类号：TP183 **文献标志码**：A **文章编号**：0529－6579（2016）04－0001－10

WASD neural network activated by bipolar sigmoid functions together with subsequent iterations

ZHANG Yunong[1,2,3], *XIAO Zhengli*[1,2,3], *DING Sitong*[1,2,3], *MAO Mingzhi*[1], *LIU Jinrong*[1]
(1. School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China;
2. Key Laboratory of Autonomous Systems and Networked Control, Ministry of Education, South China University of Technology, Guangzhou 510640, China;
3. SYSU-CMU Shunde International Joint Research Institute, Foshan 528300, China)

Abstract: A weights-and-structure-determination (WASD) algorithm is proposed for the neural network using bipolar sigmoid activation functions together with subsequent iterations, which is the combination of the Levenberg-Marquardt algorithm and the weights-direct-determination method for neural network training. The proposed algorithm, combined with the Neural Network Toolbox of MATLAB software, aims at remedying the common weaknesses of traditional artificial neural networks, such as long-time learning expenditure, difficulty in determining the network structure, and to-be-improved performance of learning and generalization. Meanwhile, the WASD algorithm has good flexibility and operability. Taking data fitting of nonlinear functions for example, numerical experiments and comparison results illustrate the superiority of the WASD algorithm for determining the optimal number and optimal weights of hidden neurons. And the resultant neural network has more excellent performance on learning and generalization.

Key words: neural networks; weights-and-structure-determination (WASD) algorithm; subsequent iterations; bipolar sigmoid activation functions; numerical experiments

* **收稿日期**：2015－07－07
基金项目：国家自然科学基金资助项目（61473323）；广州市科技计划资助项目（2014J4100057）；自主系统与网络控制教育部重点实验室开放基金资助项目（2013A07）；大学生创新创业训练计划资助项目（201410558065，201410558069）
作者简介：张雨浓（1973 年生），男；**研究方向**：人工神经网络、冗余机器人；E-mail：zhynong@mail.sysu.edu.cn

人工神经网络，简称神经网络（neural network），是模拟生物神经系统的组织结构、处理方式和系统功能的简化系统[1]。自 1980 年代形成热潮至今已近 30 年，神经网络的应用范围已扩展到许多领域[2-5]，其中，前向神经网络在各种类型的神经网络当中应用最为广泛。在现实世界中，很多非线性系统是未知的，而针对这些未知的系统进行建模是非常困难的。由于前向神经网络具有强大的非线性映射能力[6-7]，所以，可以在不清楚输入输出变量之间复杂关系的情况下，利用前向神经网络对非线性系统或函数实现建模和数据逼近[8-10]，从而为工程方面的应用提供了良好的支撑。相关研究理论已经证明[6-7,9,11-12]，一个以 S（sigmoid）函数为隐层神经元激励函数的三层前向神经网络可以实现对任何连续函数的任意精度的逼近。

学习能力和泛化能力是神经网络性能的重要反映，没有学习能力和泛化能力的神经网络是没有使用价值的[13]。值得指出的是，影响神经网络这两种能力的因素主要包括激励函数、网络结构和网络学习算法等。因此，如何选择较优的激励函数、网络结构以及网络学习算法来保证神经网络的优良性能显得非常重要[14-15]。针对前向神经网络，一些学者已进行了深入研究，并取得诸多成果[2,16-20]。具体到学习算法而言，也提出了很多有效的算法[16-20]。其中，Levenberg－Marquardt（LM）算法和广义多项式神经网络的权值直接确定法尤为突出[16-17]，它们都具有学习速度快和逼近效果好等优点，因而得到较广泛的应用[17]。在网络结构方面也有部分学者提出了一些方法和研究理论[21-23]，他们的做法能在一定程度上解决学习时间过长、网络结构难以确定、学习能力和泛化能力不足等问题，但是，就目前而言和整体而言，网络结构的确定和最优化仍然是一个挑战。

为了解决上述问题，综合考虑影响神经网络性能的主要因素，本文，在以双极 S 激励函数为隐层神经元激励函数的三层前向神经网络模型的基础上，结合两种有效的神经网络学习训练方法（也即，Levenberg-Marquardt 算法和权值直接确定法[16-17]），对隐层神经元采用边增边删的结构增长策略[18-19]，提出了一种带后续迭代的、面向双极 S 激励函数神经网络的权值与结构确定（weights-and-structure- determination，WASD）方法。该方法可以在较短时间内确定神经网络结构的同时，实现神经网络性能最优化（即，获得最好的学习能力和泛化能力）。其包括两个过程，即，确定最优隐层神经元数及最优隐层到输出层权值的过程，和输入层到隐层权值的进一步迭代优化的过程。值得指出的是，本文的三层前向神经网络模型（WASD 神经网络模型）是以双极 S 激励函数为隐层神经元激励函数的三层前向神经网络模型，其首次结合 Levenberg-Marquardt 算法和权值直接确定法，这是以往他人研究工作中没有研究过的，是本文的贡献之一。此外，在本论文中，提出了一个新颖的权值与结构确定及后续迭代的思想。即，先随机获取神经网络的输入层到隐层的权值，再用无迭代的权值直接确定法得到隐层到输出层的最优权值以及对应的最优结构，之后再利用 LM 算法反过来对输入层到隐层的权值进行后续迭代优化。也即，输入层随机化＋隐层直接确定＋输入层后续迭代。这是本文工作的另一个贡献之处。更重要的是，计算机数值实验结果证实了该算法的有效性和可行性：用该算法得到的神经网络在非线性函数数据拟合和预测方面获得优异的效果，可以达到优异的学习精度和测试精度。

1 神经网络模型

双极 S 激励函数是可微和有界的，被广泛应用于神经网络研究当中。研究表明[24-25]：使用双极 S 激励函数比使用单极 S 激励函数具有更快的收敛速度，故本文构造了一种双极 S 激励函数神经网络模型，如图 1 所示。网络的具体描述如下：输入层神经元个数设为 m，神经元阈值恒定设为 0，神经元激励函数采用线性恒等激励函数；输入层神经元到隐层神经元的连接权值在（－10，10）区间内随机初始化；隐层神经元个数设为 n，其具体个数由后述的 WASD 算法确定，神经元的阈值在（－10，10）区间内随机初始化，而神经元激励函数采用双极 S 激励函数，且其倾斜参数 c 设定为 1，即

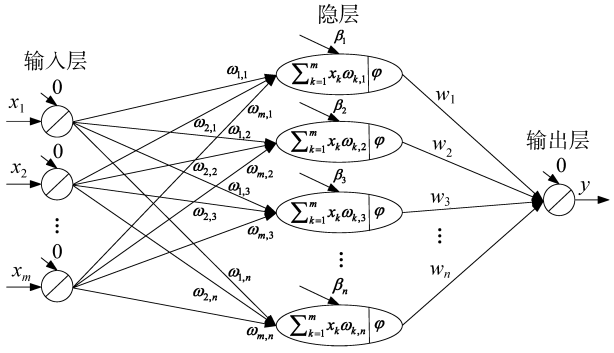


图 1 双极 S 激励函数神经网络模型

Fig. 1 Neural network model activated by bipolar sigmoid functions

$$\varphi(u) = \frac{\exp(u) - \exp(-u)}{\exp(u) + \exp(-u)} \tag{1}$$

另外，隐层神经元到输出层神经元的连接权值由后述的权值直接确定法无迭代得到；输出层神经元个数为 1，其阈值恒定设为 0，神经元激励函数也采用线性恒等激励函数。

在本文中，神经网络模型的学习样本集表示为 $\{(X_i,\gamma_i)\}_{i=1}^{N}$，其中，$N$ 为样本的个数，第 i 个样本的输入为 $X_i = [x_{i,1},x_{i,2},\cdots,x_{i,m}]^{\mathrm{T}} \in \mathbf{R}^m$，而第 i 个样本的输出，也即目标输出为 γ_i，且 $i = 1,2,\cdots,N$，上标 T 表示向量或矩阵的转置操作。更具体而言，在图 1 所示的神经网络模型中：

1）x_v 表示一个输入向量的第 v 个变量（特征），其中 $v = 1,2,\cdots,m$；

2）$\omega_{k,j}$ 为第 k 个输入层神经元和第 j 个隐层神经元之间的连接权值，其中 $k = 1,2,\cdots,m$，$j = 1,2,\cdots,n$；

3）$\sum_{k=1}^{m} x_k\omega_{k,j}$ 为第 j 个隐层神经元的输入；

4）β_j 为第 j 个隐层神经元的阈值；

5）w_j 为第 j 个隐层神经元和输出层神经元之间的连接权值；

6）y 为输出层神经元的输出值。

2　权值阈值计算

实际上，对网络进行学习训练的过程就是神经元之间的连接权值和神经元阈值的修正过程，其最终目的是：使神经网络尽量对学习样本集实现有效逼近；同时，对于学习样本集以外的测试样本或数据，神经网络也能够进行有效的辨识。尽管对神经网络权值和阈值的训练方法有很多，但并非每一种训练方法都是有效的，训练方法的优劣对网络的最终训练效果影响重大。因此，本文结合权值直接确定法和 Levenberg-Marquardt 算法作为 WASD 算法的权值和阈值计算方法。

对于图 1 所示的神经网络模型，在网络学习第 i 个样本时，第 j 个隐层神经元的输出表示为

$$p_{i,j} = \varphi\left(\sum_{k=1}^{m} x_{i,k}\omega_{k,j} - \beta_j\right)$$

输出层神经元的输出值为

$$y_i = \sum_{j=1}^{n} w_j p_{i,j}$$

神经网络的隐层神经元到输出层神经元的连接权值向量为

$$w = [w_1,w_2,\cdots,w_n]^{\mathrm{T}} \in \mathbf{R}^n$$

网络的目标输出向量表示为

$$\gamma = [\gamma_1,\gamma_2,\cdots,\gamma_N]^{\mathrm{T}} \in \mathbf{R}^N$$

对应的网络的实际输出向量为

$$y = [y_1,y_2,\cdots,y_N]^{\mathrm{T}} \in \mathbf{R}^N$$

定义网络输出均方误差（mean squared error，MSE）为

$$E = \frac{1}{N}\sum_{i=1}^{N}(\gamma_i - y_i)^2$$

对于所有的学习样本，隐层神经元的输出矩阵为

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & p_{2,3} & \cdots & p_{2,n} \\ p_{3,1} & p_{3,2} & p_{3,3} & \cdots & p_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{N,1} & p_{N,2} & p_{N,3} & \cdots & p_{N,n} \end{bmatrix} \in \mathbf{R}^{N\times n}$$

基于上述定义，根据权值直接确定法[14]，隐层神经元到输出层神经元的最优稳态权值向量可无迭代给定为

$$w = (P^{\mathrm{T}}P)^{-1}P^{\mathrm{T}}\gamma = P^{+}\gamma \tag{2}$$

其中，P^{+} 为矩阵 P 的伪逆矩阵。

基于图 1 所示的神经网络模型，在确定了网络隐层神经元到输出层神经元的最优权值以及对应的最优结构后，对输入层和隐层神经元之间的连接权值以及隐层神经元的阈值的修正，本文采用高效的 LM 迭代计算方法[15,26]。也即，在之后的计算中，将网络隐层神经元的阈值等效为网络输入层到隐层神经元的权值的一部分，具体表示如下：

$$\chi = [\chi_1,\chi_2,\cdots,\chi_M]^{\mathrm{T}} = [\omega_{1,1},\omega_{2,1},\cdots,\omega_{m,1},\omega_{1,2},\omega_{2,2},\cdots,\omega_{m,2},\cdots,\omega_{m,n},\beta_1,\beta_2,\cdots,\beta_n]^{\mathrm{T}}$$

其中 $M = nm + n$。令 $e_i(\chi) = \gamma_i - y_i(\chi)$，误差向量为 $e(\chi) = [e_1(\chi),e_2(\chi),\cdots,e_N(\chi)]^{\mathrm{T}} \in \mathbf{R}^N$。令 $J(\chi)$ 为 $e(\chi)$ 的雅克比矩阵：

$$J(\chi) = \begin{bmatrix} \frac{\partial e_1(\chi)}{\partial\chi_1} & \frac{\partial e_1(\chi)}{\partial\chi_2} & \cdots & \frac{\partial e_1(\chi)}{\partial\chi_M} \\ \frac{\partial e_2(\chi)}{\partial\chi_1} & \frac{\partial e_2(\chi)}{\partial\chi_2} & \cdots & \frac{\partial e_2(\chi)}{\partial\chi_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\chi)}{\partial\chi_1} & \frac{\partial e_N(\chi)}{\partial\chi_2} & \cdots & \frac{\partial e_N(\chi)}{\partial\chi_M} \end{bmatrix} \in \mathbf{R}^{N\times M}$$

因此，根据上述推导，基于图 1 所示的神经网络模型，输入层到隐层的连接权值以及隐层神经元的阈值可通过 Levenberg-Marquardt 算法进行如下后续迭代修正：

$$\chi_{l+1} = \chi_l + \Delta\chi_l \tag{3}$$

$$\Delta\chi_l = -[J^{\mathrm{T}}(\chi_l)J(\chi_l) + \mu_l I]^{-1}J^{\mathrm{T}}(\chi_l)e(\chi_l) \tag{4}$$

式中，l 为迭代计数（$l = 1,2,3,\cdots$），$\mu_l > 0$，I 为单

位矩阵。值得指出的是，在迭代过程中，μ_l 是一个非常重要的调整变量，其调整策略是：开始时 μ_l 取一个较小的正数（如 0.001），如果某一次调整不能使网络性能优化函数下降，则 μ_l 乘以一个大于 1 的参数 μ_{inc}（如 10），重复调整，直到使网络性能优化函数下降为止；相反，如果某一次调整可以使网络性能优化函数下降，则 μ_l 乘以一个小于 1 的参数 μ_{dec}（如 0.1）。

3 带后续迭代的 WASD 算法

神经网络的结构是影响神经网络性能最重要的因素之一，因此，神经网络的结构（也即是隐层神经元数）的选择是一个值得深入研究的问题。如果选取不当，就会影响网络的性能：具体而言，若神经元数过多，网络结构则较为冗余，网络学习与校验时因计算所需的内存空间较大，且可能出现过拟合，使网络的泛化能力下降；相反，若网络的神经元数太少，就会导致欠拟合而弱化网络的学习能力[15]。没有学习能力和泛化能力的网络是没有实用价值的[13]，因而在设计网络时，隐层神经元数的选择就极为重要。

针对隐层神经元数的选择，一些学者已进行了深入研究，但大多数的做法在实际问题求解中普遍存在着隐层神经元数庞大、学习过程振荡、学习时间过长和泛化能力较低等不足。实际上，其最终得到的网络结构是和网络的学习算法紧密相关的、非最优的[25]。以函数数据拟合为例，基于前文所述的权值直接确定算法，用图 1 所示的神经网络模型对如下非线性目标函数进行学习：

$$\gamma = x_1 \sin(0.5x_1x_2) + (0.5x_1 - 0.5)^3 + (0.5x_2 - 0.5)^3 + 5$$

学习过程中，隐层神经元数初始化设为 1，以后每次增加 1 个神经元，所得到学习误差 E 随隐层神经元数 n 递增而变化的结果如图 2 所示。

由该图可知：随着神经元数的增加，在最初时，网络学习误差下降，且下降速度较快；但当神经元数增加至一定程度，网络学习误差出现振荡，即在一个较小的范围内波动，难以继续下降，这表明，神经网络的隐神经元数开始饱和了。值得指出的是，用不同的目标函数进行实验，最终也得到与图 2 有相似特征的实验结果。

Levenberg-Marquardt 算法是神经网络 BP 算法中最优越的一个[16]，它具有训练速度快的优点，故我们可以用它来得到输入层到隐层的连接权值。结合神经网络权值直接确定法和 Levenberg-Marquardt 算法的优点[16-17]，使用边增边删的神经元增长策略[18-19]，本文提出带后续迭代的 WASD 算法。其具体描述如下。

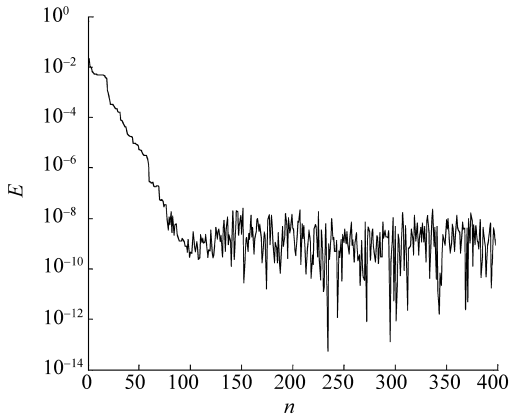


图 2 神经网络的学习误差 E 和隐层神经元数 n 的关系

Fig. 2 Relationship between neural network learning error E and neuron number n

输入：学习样本、目标精度 E_g、最长执行时间 $T_{\max}$

输出：学习误差 E、隐层神经元数 n

Step1：初始化神经网络，$E \leftarrow 10$，$n \leftarrow 1$，计数器 $C_1 \leftarrow 1$，执行时间 $T \leftarrow 0$；

Step2：输入层神经元到第 n 个隐层神经元的连接权值以及隐层神经元的阈值随机获取；

Step3：计算隐层神经元的输出矩阵 P，再根据权值直接确定法［即，公式（2）］，无迭代直接计算出最优权值 w，然后计算出当前学习误差 E_1；

Step4：若满足 $E_1 \leqslant E$，则 $E \leftarrow E_1$，增加一个隐层神经元 $n \leftarrow n+1$，$C_1 \leftarrow 1$；否则，删除第 n 个隐层神经元以及神经元的连接权值，$C_1 \leftarrow C_1 + 1$；

Step5：若 $C_1 \leqslant 5$，则跳至 Step2；否则（即，连续 5 次增加隐层神经元失败），直接进入 Step6；

Step6：计算雅克比矩阵 $J(\chi_l)$；

Step7：根据 LM 法［即，公式（4）］，计算出 $\Delta\chi_l$；

Step8：把 $\chi_l + \Delta\chi_l$ 作为变量参数，计算出当前的学习误差 E_2；

Step9：若 $E_2 \geqslant E$，则 $\mu_l \leftarrow \mu_l\mu_{\text{inc}}$，然后跳至 Step7；否则，$E \leftarrow E_2$，$\mu_l \leftarrow \mu_l\mu_{\text{dec}}$，$\chi_{l+1} \leftarrow \chi_l + \Delta\chi_l$；

Step10：在没有选择手动停止的前提下，若 $T < T_{\max}$ 且 $E > E_g$，则跳至 Step6；否则，算法结束。

显然，该算法是由以下两个过程组成：① 过程 1（Step1 至 Step5）确定最优隐层神经元数及隐层到输出层最优权值；②过程 2（Step6 至 Step10）是使用 LM 算法对输入层和隐层神经元之间的连接

权值以及隐层神经元的阈值进行后续迭代优化过程。为了更好地展示该 WASD 算法，其算法流程图如图 3 所示。

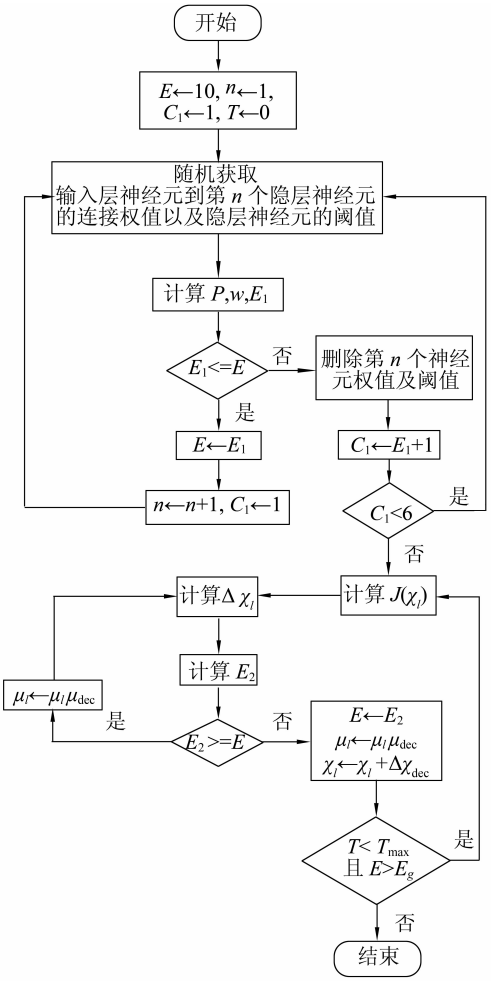


图 3　带后续迭代的 WASD 算法流程图

Fig. 3　Flowchart of WASD algorithm with subsequent iterations

值得指出的是，MATLAB 软件具有较好的开放性，除内部函数外，所有主包文件和各种工具箱都是可读、可修改的文件[27-28]。故以二输入神经网络为例，本文 WASD 算法和 Levenberg-Marquardt 算法结合调用神经网络工具箱的部分实现代码如图 4 所示。

```
T= z;                                          %学习样本处理
P=zeros (2,N);                                 %输入样本集初始化
P(1,:)=x;                                      %变量1
P(2,:)=y;                                      %变量2
net=network;                                   %生成一个神经网络对象
net.numInputs=1;                               %一个输入源
net.numLayers=2;                               %隐藏层+输出层
net.biasConnect=[1;0];                         %隐藏层有阈值，输出层没有
net.inputConnect=[1;0];                        %输出层不接受输入样本
net.layerConnect=[0 0;1 0];                    %隐层到输出层的单向传输，无自身传输
net.outputConnect=[0 1];                       %确定输出层
net.inputs{1}.range=[0 1;0 1];                 %隐层输入值的范围
net.layers{1}.size=n;                          %隐层神经元数
net.layers{1}.transferFcn='tansig';            %双极Sigmoid激励函数
Net.layers{1}.initFcn='initnw';                %输入层激励函数
net.layers{2}.size=1;                          %输出层只含一个神经元
net.layers{2}.transferFcn='purelin';           %线性激励函数
Net.layers{2}.initFcn='initnw';                %输出层默认初始化函数
Net.initFcn='initlay';                         %固定搭配
net.PerformFcn='mse';                          %性能函数为学习误差
net.trainFcn='trainlm';                        %Levenberg-Marquardt算法
Net.trainParam.show=50;                        %学习50次刷新一次图像
net.trainParam.epochs=Inf;                     %最大迭代次数
net.trainParam.time=T_max-T_1;                 %T_1为迭代前的训练时间
net.trainParam.goal=goal;                      %目标精度
net.trainParam.mu=0.001;                       %迭代初始值
net.trainParam.mu_inc=10;                      %LM的增大因子
net.trainParam.mu_dec=0.1;                     %LM的缩减因子
net.layerWeights{2,1}.learn=0;                 %隐层到输出层的连接权值不迭代
net=init(net);                                 %网络初始化
Net.iw{1,1}=weight_input2hidden;               %输入层到隐层的连接权值
Net.lw{2,1}=weight_hidden2output;              %隐层到输出层的连接权值
net.b{1}=bIas;                                 %隐层阈值
[net,tr]=train(net,P,T);                       %开始训练
```

图 4　二输入 WASD 神经网络涉及工具箱调用的部分实现代码

Fig. 4　Partial code of implementation of two-input WASD neural network involving toolbox calling

4　WASD 网络逼近能力分析

对一个三层的神经网络，许多学者证明了如下的逼近定理[6,11-12]。

定理 1　令 $\varphi(\cdot)$ 为有界的单调递增连续函数，H_m 代表 m 维单位超立方体 $[0,1]^m$。$C(H_m)$ 表示定义在 H_m 上的连续函数构成的集合，则给定任何目标函数 $\gamma\in C(H_m)$ 和目标精度 $\varepsilon>0$，存在整数 n 和实常数组 $\{w_j\}$、$\{\beta_j\}$ 和 $\{\omega_{k,j}\}$，其中 $k=1,2,\cdots,m$，$j=1,2,\cdots,n$，使得网络输出

$$y(x_1,x_2,\cdots,x_m)=\sum_{j=1}^{n}w_j\varphi\Big(\sum_{k=1}^{m}\omega_{k,j}x_k-\beta_j\Big)$$

可以以任意精度逼近目标函数 $\gamma(x_1,x_2,\cdots,x_m)$，即

$$\big|y(x_1,x_2,\cdots,x_m)-\gamma(x_1,x_2,\cdots,x_m)\big|<\varepsilon,$$
$$\forall(x_1,x_2,\cdots,x_m)\in H_m$$

对照和推广上述定理，双极 S 激励函数为有界的单调递增连续函数，因此可得该激励神经网络能以理论任意精度逼近非线性连续函数。而“边增边删”实际上是在增删神经元的过程中使得神经网络的隐层神经元数不断地逼近神经网络结构最优时的数目。所以，理论而言，当增删神经元的次数趋向于无穷时（实际而言，达到某个较大次数时），我们可以找到使得神经网络结构最优的隐层神经元数（即，定理 1 中的 n 值）。而且，作者在之前的工作中已做过大量的数值实验。实验结果显示：随着隐层神经元个数的增加，网络的测试误差呈现一个“V”或“U”字形的图像[29-30]。因此，利用“边增边删”方法可以找到最优隐层神经元数。

5 计算机数值实验验证

为了验证本文提出的 WASD 神经网络模型的性能，本文以 2 输入和 3 输入的数据拟合为例，分别对以下目标函数（5）－（7）进行函数数据拟合实验。

$$\gamma = [0.5((x_1-1)^3+(x_2-1)^3)\cdot \cos(0.5x_1+x_2)]/\exp(-x_1-x_2+0.2)+5 \tag{5}$$

$$\gamma = \frac{2\sin \boldsymbol{A}}{\boldsymbol{A}}+10 \tag{6}$$

$$\boldsymbol{A} = \sqrt{(15(x_1-0.5))^2+(15(x_2-0.5))^2+2};$$

$$\gamma = 0.5x_1x_2x_3\exp(-x_1^5-x_2^3-x_3^4)+2 \tag{7}$$

其中，数据产生方法如下：对于目标函数（5）和（6），在区间 $[0, 1]^2$ 内（每个维度采样间隔为 0.05）均匀产生 441 个学习样本，在区间 $[0.05, 0.95]^2$ 内（每个维度采样间隔为 0.017）均匀产生 2 809 个测试样本；对于目标函数（6），还在区间 $[0.05, 1.04]^2$ 内（每个维度采样间隔为 0.017）均匀产生 3 249 个测试和预测样本；对于目标函数（7），在区间 $[0.1, 0.8]^3$ 内（每个维度采样间隔为 0.01）均匀产生 512 个学习样本，在区间 $[0.14, 0.76]^3$ 内（每个维度采样间隔为 0.017）均匀产生 50 653 个测试样本。另外指出的是，进行该批 MATLAB 7.6.0 数值实验的计算机的硬件配置为 Intel（R）Core（TM）2 Duo CPU（主频 2.2 GHz）和 2.0 GB 内存。而使用本文的 WASD 算法对目标函数（5），（6）和（7）进行函数数据拟合，所得到实验结果（5 次实验取平均值）如表 1 所示。

从表 1 可以看出：通过 WASD 算法训练得到的 WASD 神经网络，学习误差和测试误差均可达到非常小的数量级，网络具有优异的学习能力和泛化能力，对非线性函数数据的逼近效果理想；而且算法执行时间较短。为了和 WASD 算法进行比较，本文也分别单独使用 Levenberg－Marquardt 算法和文献［20］中所提出的算法（即，LMFNN 算法）进行对比性数值实验。由于单独的 LM 算法需要提前给定隐层神经元数，但是，对于不同的训练样本，网络的最优隐层神经元数是不同的，因此我们在对比实验中分别选择了 25、50、100 和 200 个隐层神经元进行实验。此外，本文使用的是只有一层隐含层的神经网络结构。根据文献［20］的算法，对目标函数（5）和（6），可求得其隐含层神经元数为 110；对目标函数（7），其隐含层神经元数为 128。为了便于实验对比，在迭代过程中均采用手动停止的操作。Levenberg-Marquardt 算法和 LMFNN 算法的数值实验结果（5 次实验取平均值）分别在表 2 和表 3 中给出。

从表 2 可以看出：①当隐层神经元数较少时，通过 LM 算法训练得到的神经网络（简称 LM 神经网络）的学习误差和校验误差相对较大，其学习能力和泛化能力一般表现出轻微的欠拟合现象，此外，网络的学习时间相对较长；②当隐层神经元数进一步增加时，LM 神经网络的学习误差和测试误差变得较小，具有较好学习能力和泛化能力，然而，网络的学习时间也稍长；③当隐层神经元数较多时，LM 神经网络的学习误差较小，但校验误差较大，两者不一致，反映出 LM 网络的学习能力较好而泛化能力却差的过拟合现象。结合上述分析，可以大概找到 LM 神经网络的较优隐层神经元数，即学习能力和泛化能力均较优的 LM 神经网络结构。另外，从表 3 可以看出：相比于 LM 网络，LMFNN 算法的学习误差和测试误差均处在较小的数量级，且执行时间也相对较短，能得到较优神经网络结构，能实现对目标函数的有效逼近。

综合表 1－3 可知，无论在网络性能还是运行时间方面，WASD 算法均优于单独使用 LM 算法或者 LMFNN 算法。而且，相对于单独使用 LM 算法而言，WASD 算法可以自动确定最优隐层神经元数，单独的 LM 算法则需要提前设定隐层神经元数（其一般难以达到最优）。所以，如果要确定 LM 神经网络的最优隐层神经元数，以试错法形式的反复测试校正将会耗费非常长的计算时间。而对比于 LMFNN 算法（其测试误差较低），WASD 算法能得到更优的隐含层神经元数、网络结构和性能。

表 1 WASD 神经网络的学习与测试结果

Table 1 Learning and testing results of WASD neural network

目标函数	最优隐层神经元数/个	学习误差	测试误差	执行时间/s
(5)	106	1.215×10^{-9}	1.164×10^{-9}	9.405
(6)	202	3.255×10^{-6}	3.712×10^{-6}	9.244
(7)	209	8.552×10^{-10}	2.607×10^{-9}	40.180

表 2　用于对比的 LM 神经网络学习与测试结果

Table 2　Learning and testing results of comparative LM neural network

目标函数	隐层神经元数/个	学习误差	测试误差	执行时间/s	总执行时间/s
(5)	25	1.741×10^{-8}	1.441×10^{-8}	12.494	49.107
	50	5.318×10^{-9}	7.640×10^{-9}	12.670	
	100	9.467×10^{-9}	2.092×10^{-7}	12.500	
	200	2.028×10^{-7}	4.772×10^{-5}	11.443	
(6)	25	6.033×10^{-5}	4.943×10^{-5}	12.391	48.173
	50	6.751×10^{-6}	9.782×10^{-5}	12.337	
	100	2.683×10^{-6}	2.525×10^{-9}	11.600	
	200	1.604×10^{-6}	1.300×10^{-3}	11.845	
(7)	25	1.712×10^{-8}	1.558×10^{-8}	11.412	40.519
	50	1.885×10^{-9}	4.821×10^{-9}	9.083	
	100	1.069×10^{-8}	1.812×10^{-7}	10.768	
	200	2.348×10^{-9}	5.383×10^{-7}	9.256	

表 3　用于对比的 LMFNN 神经网络学习与测试结果

Table 3　Learning and testing results of comparative LMFNN neural network

目标函数	隐层神经元数/个	学习误差	测试误差	执行时间/s
(5)	110	4.301×10^{-8}	1.786×10^{-6}	13.606
(6)	110	2.369×10^{-6}	1.324×10^{-4}	11.415
(7)	102	1.423×10^{-9}	1.163×10^{-7}	39.063

为了更进一步地验证和展示基于双极 S 激励函数神经网络的 WASD 算法的有效性，下面以目标函数（6）为例（设定执行时间为 100 s），进行数值实验，实验结果如图 5 至图 9 所示。

具体而言，图 5 显示了 WASD 算法在两个学习过程当中，网络学习误差的变化情况。在过程 1（Process 1）中，随着神经元数的增加，网络的学习误差逐渐降低，由于采用了边增边删的策略，期间并无发生振荡现象，最终确定的隐层神经元数为 207；进入过程 2（Process 2）后，网络的学习误差一直下降，直至程序（达到结束条件）结束，最终训练停止时，网络的学习误差到达 8.455×10^{-7}。图 6 是神经网络对目标函数（6）的学习结果和校验结果。此外，图 7 和图 8 展示了 WASD 神

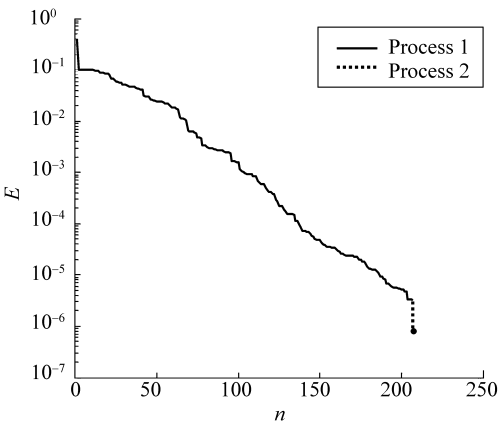
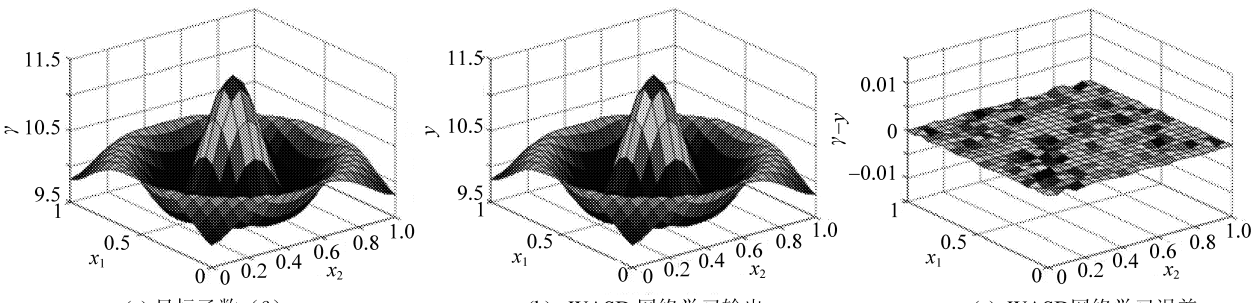


图 5　WASD 神经网络的学习误差 E 和神经元数 n 的关系

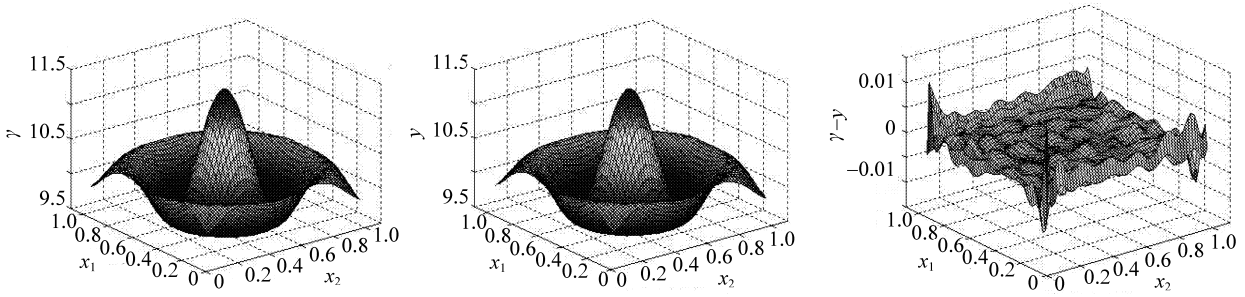
Fig. 5　Relationship between WASD neural network learning error E and neuron number n



(a) 目标函数（6）　(b) WASD 网络学习输出　(c) WASD网络学习误差

图 6　WASD 神经网络对目标函数（6）在区间 $[0, 1]^2$ 内的学习结果

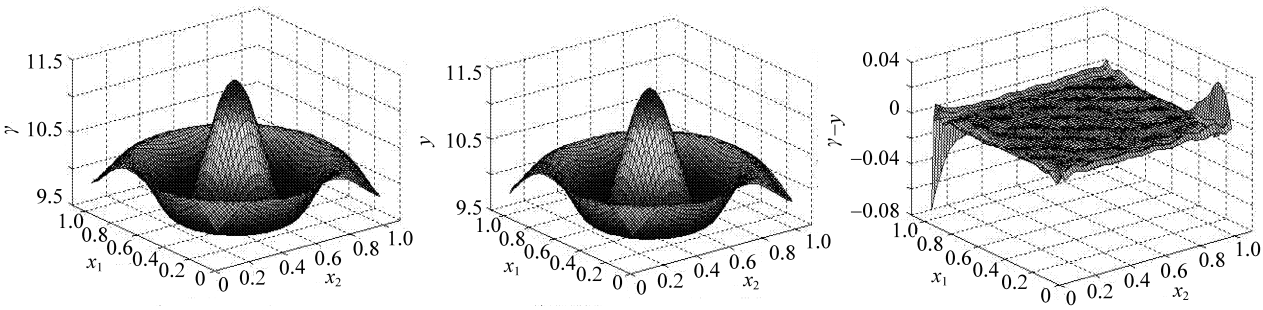
Fig. 6　Learning results of WASD neural network for target function (6) in $[0, 1]^2$



(a) 目标函数(6) (b) WASD网络测试和预测输出 (c) WASD网络输出误差

图 7 WASD 神经网络对目标函数（6）在区间 $[0.05, 1.01]^2$ 内的测试和预测结果

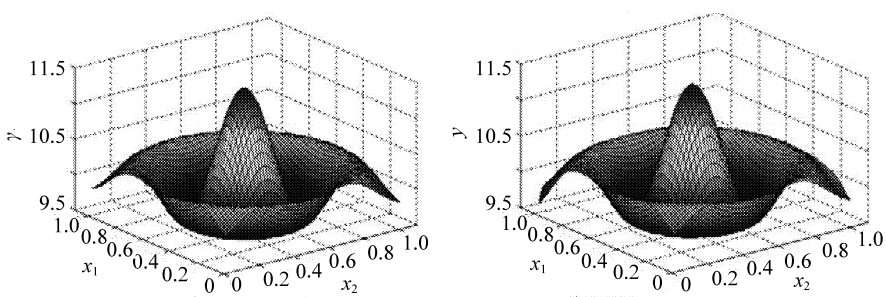
Fig. 7 Testing and forecasting results of WASD neural network for target function (6) in $[0.05, 1.01]^2$



(a) 目标函数(6) (b) WASD网络测试和预测输出 (c) WASD网络输出误差

图 8 WASD 神经网络对目标函数（6）在区间 $[0.05, 1.02]^2$ 内的测试和预测结果

Fig. 8 Testing and forecasting results of WASD neural network for target function (6) in $[0.05, 1.02]^2$



(a) 目标函数(6) (b) WASD网络测试和预测输出

图 9 WASD 神经网络对目标函数（6）在区间 $[0.05, 1.04]^2$ 内的测试和预测结果

Fig. 9 Testing and forecasting results of WASD neural network for target function (6) in $[0.05, 1.04]^2$

经网络对目标函数（6）在 $[0.05, 1.02]^2$ 内的测试和预测结果。可以看出，此时神经网络的测试和预测误差均较小，尤其是在 $[0.05, 1.01]^2$ 内的测试和预测误差小于 0.015。更进一步地，图 9 显示了 WASD 神经网络对目标函数（6）在 $[0.05, 1.04]^2$ 内的测试和预测结果。虽然网络输出与目标输出函数之间存在差别，但结果还是令人满意的。从上述结果可知，通过该算法得到的 WASD 神经网络具有较好的泛化能力。

综上所述，WASD 算法可以根据已有的样本数据，通过网络的学习训练，无需人工干预，自动地确定出最优隐层神经元数，同时实现对权值和阈值的后续有效调整，而且执行时间较短，最终得到的 WASD 网络具有很好的学习能力和泛化能力，能够对非线性函数数据进行有效的拟合。相比较而言，单独 LM 神经网络的隐层神经元数需要人为提前设定，如果想要获得较好的效果和性能，需要反复地进行尝试或者根据设计者的经验进行隐层神经元数的人为优化，过程相当耗时。而 LMFNN 算法，虽然可以自动确定隐层神经元数和网络结构，但由于

其算法受到样本大小影响，当学习样本的数量和分布不够合理时，所得到的隐层神经元数和网络结构未必保证是最优。

此外，为了验证 WASD 算法中 LM 方法的有效性和优越性，针对目标函数（5）、（6）和（7），使用不同的训练方法（即，在后续迭代中采用 LM 算法）来优化输入层和隐层神经元的连接权值以及隐层神经元的阈值。表 4 为对应的数值结果（取 5 次实验平均值）。

表 4　不同训练方法的性能对比

Table 4　Performance comparisons among different training methods

目标函数	误差类型	LM	动量 BP	BFGS	最速下降 BP 法
(5)	学习误差	1.822×10^{-9}	2.900×10^{-8}	1.179×10^{-8}	4.149×10^{-8}
	测试误差	1.803×10^{-9}	1.3004×10^{-8}	3.397×10^{-8}	3.214×10^{-8}
(6)	学习误差	2.075×10^{-6}	1.979×10^{-6}	8.751×10^{-5}	6.013×10^{-6}
	测试误差	2.417×10^{-6}	1.997×10^{-6}	8.510×10^{-5}	9.148×10^{-6}
(7)	学习误差	8.640×10^{-10}	6.911×10^{-8}	1.300×10^{-8}	6.278×10^{-8}
	测试误差	5.653×10^{-9}	9.811×10^{-8}	3.366×10^{-8}	7.372×10^{-8}

从表 4 中可以看出：LM 方法的训练误差和测试误差总体而言优于其它的训练方法（如 BFGS 和最速下降法）。此结果说明了 LM 方法在 WASD 算法中的有效性及优越性。此外，针对目标函数(5)，本文使用 WASD 算法进行 50 次实验，其目标精度设置为 5.0×10^{-12}，执行时间为 20 s，表 5 为 50 次实验的统计结果。

表 5　WASD 算法 50 次实验的统计结果

Table 5　Statistical results of WASD algorithm in 50 experiments

误差类型	均值	标准差
学习误差	1.639×10^{-9}	1.557×10^{-9}
测试误差	1.747×10^{-9}	1.708×10^{-9}

从表 5 中可以看出：WASD 算法的学习误差与测试误差的均值和标准差都处于 10^{-9}这一微小量级。此结果（特别是其标准差处于较小的量级）说明了 WASD 算法具有较好的稳定性和收敛性。

值得指出的是，我们还在神经网络算法性能实验中采用了交叉验证的评价方法。具体而言，针对目标函数（5）进行了 10 折（10 - fold）交叉验证。在 10 次验证中，学习误差最小为 2.000×10^{-12}，此时的测试误差为 5.512×10^{-10}；测试误差最小为 3.797×10^{-10}，此时学习误差为 9.793×10^{-12}。总体而言，10 折交叉验证学习误差的均值为 2.349×10^{-10}，测试误差的均值为 2.304×10^{-7}。可以看出，本文提出的算法在多重交叉验证实验中，学习误差和测试误差也均处于较小的量级。该结果进一步证实了所提出算法的有效性和优越性。

6　结　语

本文首先构造了一种三层的双极 S 激励函数前向神经网络模型，进而提出了一种带后续迭代的权值与结构确定算法。为了实现在较短时间内确定最优隐层神经元数的同时，使神经网络获得更好的学习性能和泛化性能，该算法创造性地结合了权值直接确定法与 Levenberg-Marquardt 算法各自的优点，并采用了边增边删的结构增长策略。计算机数值实验和对比结果表明：基于 WASD 算法的双极 S 激励函数神经网络具有更为优异的学习能力和泛化能力，学习时间少（或适中），在非线性函数数据拟合方面可表现出很好的效果。本文提出的带后续迭代的神经网络权值与结构确定算法因此可具有广泛的应用前景（其限于篇幅不再赘述）。而如何在更进一步的实际应用中发现本算法的不足和做出相应的改进措施或也是下一步工作的重点和未来研究方向。

参考文献：

[1] 张青贵. 人工神经网络导论[M]. 北京：中国水利水电出版社，2004.

[2] 张雨浓，王茹，劳稳超，等. 符号函数激励的 WASD 神经网络与 XOR 应用[J]. 中山大学学报(自然科学版)，2014，53(1)：1 - 7.

[3] SHRIVASTAVA S，SINGH M P. Performance evaluation of feed-forward neural network with soft computing techniques for hand written English alphabets [J]. Applied Soft Computing，2011，11(1)：1156 - 1182.

[4] 杨文光. 权值直接确定的三角型模糊前向神经网络

[J]. 中山大学学报(自然科学版), 2013, 52(2): 33 – 37.

[5] 蔡娟,蔡坚勇, 廖晓东,等. 基于卷积神经网络的手势识别初探[J]. 计算机系统应用, 2015, 24(4): 113 – 117.

[6] HORNIK K, STINCHCOMBE M, WHITE H. Multilayer feedforward networks are universal approximators [J]. Neural Networks, 1989, 2(5): 359 – 366.

[7] CHEN T, CHEN H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems [J]. IEEE Transactions on Neural Networks, 1995, 6(4): 911 – 917.

[8] GUERRA F A, COELHO L S. Multi-step ahead nonlinear identification of Lorenz's chaotic system using radial basis neural network with learning by clustering and particle swarm optimization [J]. Chaos Solitons and Fractals, 2008, 35(5): 967 – 979.

[9] 王建军,徐宗本. 多元多项式函数的三层前向神经网络逼近方法[J]. 计算机学报, 2009, 32(12): 2482 – 2488.

[10] 李凯,翟振华. 神经网络在函数逼近问题中的应用研究[J]. 计算机工程, 2001, 27(5): 189 – 190.

[11] COTTER N E. The Stone-Weierstrass theorem and its application to neural networks [J]. IEEE Transaction on Neural Networks, 1990, 1(4): 290 – 295.

[12] CYBENKO G. Approximation by superposition of a sigmoidal function [J]. Mathematics of Control, Signals, Systems, 1989, 2(4): 303 – 314.

[13] 魏海坤,徐嗣鑫,宋文忠. 神经网络的泛化理论和泛化方法[J]. 自动化学报, 2001, 27(6): 806 – 815.

[14] JAIN A K, MAO J C, MOHIUDDIN K M. Artificial neural networks: a tutorial [J]. Computer, 1996, 29(3): 31 – 34.

[15] LAWRENCE S, GILES C L, TSOI A C. What size neural network gives optimal generalization? Convergence properties of backpropagation, UMIACS-TR-96 – 22 and CS-TR – 3617 [R]. College Park: University of Maryland College Park, 1996.

[16] 苏高利,邓芳萍. 论基于 MATLAB 语言的 BP 神经网络的改进算法[J]. 科技通报, 2003, 19(2): 130 – 135.

[17] 张雨浓,杨逸文,李巍. 神经网络权值直接确定法[M]. 广州: 中山大学出版社, 2012.

[18] 张雨浓,罗飞恒,陈锦浩,等. 三输入伯努利神经网络权值与结构双确定[J]. 计算机工程与科学, 2013, 35(05): 142 – 148.

[19] ZHANG Y, DING S, LIU X, et al. WASP neuronet activated by bipolar-sigmoid functions and applied to glomerular-filtration-rate estimation [C]. The 26th Chinese Control and Decision Conference (CCDC), IEEE, 2014: 172 – 177.

[20] 李鸿儒,王晓楠,何大阔,等. 一种优化计算确定神经网络结构的方法[J]. 系统仿真学报, 2009, 21(1): 104 – 107.

[21] BAHI J M, CONTASSOT-VIVIER S, SAUGET M. An incremental learning algorithm for function approximation [J]. Advance in Engineering Software, 2009, 40(8): 725 – 730.

[22] BENARDOS P G, VOSNIAKOS G C. Optimizing feedforward artificial neural network architecture [J]. Engineering Applications of Artificial Intelligence, 2007, 20(4): 365 – 382.

[23] TSAI J T, CHOU J H, LIU T K. Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm [J]. IEEE Transactions on Neural Networks, 2006, 17(1): 69 – 80.

[24] ARULAMPALAM G, RAMAKONAR V, BOUZERDO A, et al. Classification of digital modulation schemes using neural networks [C]. The 5th International Symposium on Signal Processing and Its Applications (ISSPA), IEEE, 1999, 2: 649 – 652.

[25] HAYKIN S. Neural networks: a comprehensive foundation [M]. 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 2004.

[26] HAGAN M, DEMUTH H, BEALE M. Neural network design [M]. Beijing: China Machine Press, 2002.

[27] 傅荟璇,赵红. MATLAB 神经网络应用设计[M]. 北京: 机械工业出版社, 2010.

[28] DEMUTH H, BEALE M, HAGAN M. Neural network toolbox 6 user's guide [M]. The MathWorks Press, 2009.

[29] ZHANG Y, YU X, GUO D, et al. Weights and structure determination of multiple-input feed-forward neural network activated by Chebyshev polynomials of class 2 via cross-validation [J]. Neural Computing and Applications, 2014, 25(7/8): 1761 – 1770.

[30] ZHANG Y, YIN Y, GUO D, et al. Cross-validation based weights and structure determination of Chebyshev polynomial neural networks for pattern classification [J]. Pattern Recognition, 2014, 47: 3414 – 3428.