Article ID: 1000 - 8152(2005)04 - 0578 - 05

Chaos optimization algorithm design for fuzzy neural network

ZOU En[1,2], LI Xiang-fei[1], ZHANG Tai-shan[2]

(1. Department of Electrical Engineering, Zhuzhou Institute of Technology, Zhuzhou Hunan 412008, China;
2. Information Science & Engineering Institute, Central South University, Changsha Hunan 410083, China)

Abstract: An optimization algorithm design based on chaotic variable is proposed for multilayer fuzzy neural network. Off-line optimization uses chaos algorithm and chaos variables are applied to search for network structure and parameters, in which the network is in dynamic chaos state. An approximate optimal network structure and parameters are found from dynamic network according to performance index. On-line optimization uses gradient descent algorithm and the initial values of gradient descent searching are parameters approximately global optimal values from chaos searching, the parameters of fuzzy neural network are further adjusted. The global optimal values of network are searched quickly by means of combination of chaos global searching and gradient descent local searching. Finally, second order delay system is simulated, and the results show that the chaos optimal control is of high precision, small overshoot, fast response and good robustness.

Key words: fuzzy neural network; optimization; chaotic variables; gradient descent algorithm

CLC number: TP273 **Document code**: A

模糊神经网络的混沌优化算法设计

邹 恩[1,2], 李祥飞[1], 张泰山[2]

(1. 株洲工学院 电气工程系, 湖南 株洲 412008; 2. 中南大学 信息科学与工程学院, 湖南 长沙 410083)

摘要: 提出了一种基于混沌变量的多层模糊神经网络优化算法设计. 离线优化部分采用混沌算法, 将混沌变量引入到模糊神经网络结构和参数的优化搜索中, 使整个网络处于动态混沌状态, 根据性能指标在动态模糊神经网络中寻找较优的网络结构和参数. 在线优化部分采用梯度下降法, 把混沌搜索后得到的参数全局次优值作为梯度下降搜索的初始值, 进一步调整模糊神经网络的参数, 实现混沌粗搜索和梯度下降细搜索相结合的优化目的, 能较快地找到全局最优解. 最后对二阶延迟系统进行仿真, 结果表明混沌优化方法控制精度高、超调小、响应快和鲁棒性强.

关键词: 模糊神经网络; 优化; 混沌变量; 梯度下降法

1 Introduction

Fuzzy neural networks (FNN) have integrated fuzzy logic inference capability of fuzzy systems and adaptive capability of neural networks, it inherits the advantages of both and has drawn the attention and wide research of many researchers[1,2]. To seek an optimal FNN, a number of exact methods have been proposed so far to deal with the combinatorial optimization problems, such as BP algorithm, gradient descent method, and so on, but they have disadvantages of slow convergence and the tendency to become trapped in local minimum if initial values are not suitably selected. Simulate annealing method has been widely applied to various optimization problems too[3], but it requires subtle adjustment of parameters in the annealing schedule such as the size of the temperature steps during annealing, the temperature range, the number of re-starts and re-direction of the search, etc. It is very difficult to design a detailed optimal FNN system with the methods mentioned above.

Chaos optimization method has many advantages. It can search approximate global optimal solution[4], but if chaos optimization is adopted solely, the learning speed from approximate global optimal solution to global optimal solution is slow. as compared with gradient descent method. In view of convergence rate and convergence accuracy, a two-phase optimization scheme is proposed in the paper. In phase one, chaos off-line optimization scheme is used for global optimization parameters of membership function and weights as well as the structure of FNN, and gets an approximate global optimal FNN. In phase two, gradient descent on-line optimization scheme is used to locally adjusting FNN's parameters for desired outputs, which can converge quickly to global optimal

Received date: 2002 - 12 - 03; Revised date: 2004 - 12 - 08.

Foundation item: supported by Hunan Province Natural Science Foundation (01JJY3029); China National Packaging Corporation Science and Technology Foundation (05ZBKJA003).

value.

2 Fuzzy neural network controller

First, an initial form of the FNN is constructed, for simplicity, we suppose that the network has four layers, the structure of FNN is shown in Fig. 1.

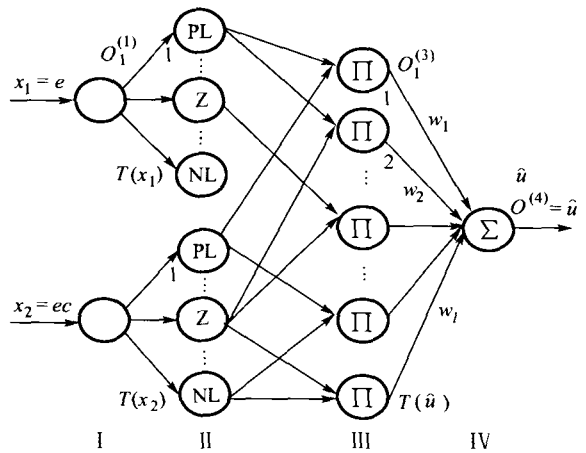


Fig. 1 Structure of fuzzy neural network

Layer I ~ III realize "if-then" rule of fuzzy control, the Layer IV realizes defuzzification computation, each layer performing one stage of the fuzzy inference process is described as follows: .

Layer I is the input layer in which the nodes transmit input values directly to the corresponding nodes of the next layer without any computation. The number of nodes in Layer I equals that of the normalized input variables, the input and output are shown as follows:

$$O_i^{(1)} = x_i = e \text{ or } ec,\ x_i \in [-1,1],\ i = 1,2. \tag{2.1}$$

Layer II is membership function layer in which each node represents a fuzzy subset of variables $T(x_i)(i = 1,2)$, i.e. NL, Z or PS, the total number of nodes in this layer is $S = T(x_1) + T(x_2)$, the output of each node simulates Gaussian function as membership function as follows:

$$O_j^{(2)} = \mu_{F_{ij}} = \exp\left[-\frac{(w_{ij}^{(2)}O_i^{(1)} - \theta_{ij})^2}{\sigma_{ij}^2}\right] = \exp\left[-\frac{(x_i - \theta_{ij})^2}{\sigma_{ij}^2}\right],\ i = 1,2,\cdots,S, \tag{2.2}$$

where θ_{ij} and σ_{ij} are center and width of the membership function of the jth term in the ith input, respectively, while $(-1 < \theta < 1), (0 < \sigma < 1)$.

Layer III is fuzzy rule layer in which the links are used for performing antecedent matching of fuzzy logic rules. The total number of nodes in Layer III equals the number of fuzzy control regulation, i.e. $F = T(x_1) \times T(x_2)$, each node represents a fuzzy logic rule as follows:

$$O_k^{(3)} = \prod_{i=1}^{2} w_{jk}^{(3)} O_j^{(2)} = \mu_{F_{ij}}(x_1) \cdot \mu_{F_{ij}}(x_2),\ k = 1,2,\cdots,F. \tag{2.3}$$

Layer IV is defuzzification layer in which all of the rule nodes are linked with output, the number of nodes equal the number of output variables that have been normalized, and they are represented as follows:

$$\begin{cases} O^{(4)} = \hat{u} = \dfrac{\sum_l w_{kl}^{(4)} O_k^{(3)}}{\sum_l O_k^{(3)}} = \dfrac{\sum_l w_{kl}^{(4)} (\prod_i \mu_{F_{ij}}(x_i))}{\sum_l (\prod_i \mu_{F_{ij}}(x_i))} = \\ \dfrac{\sum_l w_{kl}^{(4)} (\prod_i \exp\left(-\left(\frac{x_i - \theta_{ij}}{\sigma_{ij}}\right)^2\right))}{\sum_l (\prod_i \exp\left(-\left(\frac{x_i - \theta_{ij}}{\sigma_{ij}}\right)^2\right))}, \\ O^{(4)} \in [-1,1],\ l = 1, \end{cases} \tag{2.4}$$

the link weights equal to 1 except for layer (IV).

3 Optimization design method of FNN

The process of optimal design consists of chaos off-line optimization and gradient descent on-line optimization, which is discussed in the following section.

3.1 Chaos off-line optimize structure and parameters of the networks

From Fig. 1, the FNN structure discussed above shows that the parameters of FNN will realize optimization after modification of the centers and widths of Layer II and the link weights of Layer IV. The total number of parameters of centers θ_{ij} and widths σ_{ij} in Layer II are $2S$, and that of weights parameters of Layer IV are F, therefore, the total number of parameters to be optimized should be

$$M = 2S + F = 2(T(x_1) + T(x_2)) + T(x_1) \times T(x_2). \tag{3.1}$$

θ, σ and w could satisfy constraint condition

$$-1 < \theta,\ w < 1 \text{ and } 0 < \sigma < 1. \tag{3.2}$$

Because the link weights at Layer IV represent the existence strength of the corresponding rule, the structure optimization modifies the links between the rule nodes and output nodes, and the number of rule nodes. The greatest number of rule nodes should be

$$n_{\max} = F = T(x_1) \times T(x_2). \tag{3.3}$$

As the link of rule node with weight of Layer IV is one-one correspondences, pruning method, that is called destructive method, is adopted to structure optimization. Some weights of this initial network decrease little by little

and tends to the neighborhood of zero during the learning of weight parameters, the corresponding links are deleted step by step since it does not affect the outputs, and redundant nodes are deleted too[5].

When the efficient nodes are searched out from the all of rule nodes in Eq. (3.1), redundant nodes are eliminated but precision of system must be assured. In view of the precision of control and simple property of structure, the penalty factor expressing the complexity of network structure is brought forward in performance index as follows:

$$J_1 = \min(E + \lambda C), \tag{3.4}$$

where
$$E = \frac{1}{2}\sum_{k=1}^{N}(r(k) - y(k)), \tag{3.5}$$

and
$$C = \sum_{i=1}^{n_{\max}} \frac{(w_i/\alpha)^2}{1 + (w_i/\alpha)^2} \tag{3.6}$$

is punish item, $0.15 > \alpha \geqslant 0, 0.5 > \lambda > 0$.

In the learning process, if $|w_i| \leqslant \alpha$, then the node connecting w_i and w_i itself are deleted, otherwise, the node and w_i are retained.

Consider the famous Logistic mapping

$$Q_{n+1} = 4Q_n(1-Q_n),\ n = 0,1,2,\cdots,N,\ Q_0 \in (0,1), \tag{3.7}$$

where $Q = (q_{1,n}, q_{2,n}, \cdots, q_{L,n})$ are the numbers of chaotic variables, $L = M, N$ is the frequency of chaos learning. In order that each chaos variable can map to itself interval, mapping of chaos variables can be described as follows:

$$q^*_{i,n} = 2q_{i,n} - 1 \quad (i = 1,2,\cdots,S,\cdots,S+F, n = 1,2,\cdots,N), \tag{3.8}$$

$$q^*_{j,n} = q_{j,n}(j = i+1,\cdots,M, n = 1,2,\cdots,N), \tag{3.9}$$

where $q^*_{i,n}$ are parameters of membership function's weights and center, the equation (3.8) can ensure $-1 < (w,\theta) < 1$, $q^*_{j,n}$ are parameters of membership function's width, the equation (3.9) can ensure $0 < \sigma < 1$. Then, the chaos variables of mapping are carried-wave and iterated respectively, the algorithm is as follows:

1) The M different initial values are sampled at random in (0,1) interval and are substituted in Eq.(3.7), each parameter corresponds to a chaos variable.

2) Assume each variable satisfies constraint condition (3.2), the $2S$ chaos variables are substituted in Eq.(3.8) and the F chaos variables are substituted in Eq.(3.9).

3) The M chaotic variables are substituted in Eqs.(2.1) ~ (2.4) and start to search for global approximate optimal values $\theta^{**}, \sigma^{**}, w^{**}$.

4) If the nodes are found out to be connecting with $|w_i^{**}| \leqslant \alpha, i \in [1, n_{\max}]$, then the nodes and w_i^{**} itself are deleted altogether, θ^{**}, σ^{**} and residual w^{**} are reserved.

5) According to Eqs.(3.4) ~ (3.6) compute performance index J_1, if it coincides with the requirement condition, iteration is stopped and results are output, otherwise, Step 3) is repeated.

The FNN is connected with system when training is finished.

3.2 Gradient descent on-line optimization algorithm

Chaotic motion has features of ergodicity, but it requires a long time to reach the global optimal. When global approximate optimal values have searched out with chaotic optimization method, the network enters the second learning phase to on-line adjust parameters with gradient descent algorithm, the approximate global optimal values of chaos searching are the initial values of gradient descent searching, the system can reach global optimum solutions quickly.

The performance index of gradient descent optimization is as follows:

$$J_2 = \frac{1}{2}(r(t) - y(t))^2 = \frac{1}{2}e(t)^2, \tag{3.10}$$

where $e(t)$ is the system error.

In Eq.(2.4), let

$$\begin{cases} a = \prod_{i=1}^{2} \exp\left(-\left(\frac{x_i - \theta_{ij}}{\sigma_{ij}}\right)^2\right), \\ b = \sum_l w_{kl}a,\ c = \sum_l a,\ \hat{u} = \frac{b}{c}. \end{cases} \tag{3.11}$$

According to Eq.(3.10) and Eq.(3.11)

$$\frac{\partial J_2}{\partial w_{kl}} = e(t)\frac{\partial e(t)}{\partial w_{kl}} = -e(t)\frac{\partial y(t)}{\partial w_{kl}} = -e(t)\frac{\partial y(t)}{\partial \hat{u}(t)}\frac{\partial \hat{u}(t)}{\partial w_{kl}} = -e(t)\cdot y_{\hat{u}}(t)\cdot\frac{\partial \hat{u}(t)}{\partial b}\frac{\partial b}{\partial w_{kl}} = -e(t)\cdot y_{\hat{u}}(t)\cdot\frac{1}{c}\cdot a, \tag{3.12}$$

$$\frac{\partial J_2}{\partial \theta_{ij}} = e(t)\frac{\partial e(t)}{\partial \theta_{ij}} = -e(t)\frac{\partial y(t)}{\partial \theta_{ij}} = -e(t)\frac{\partial y(t)}{\partial \hat{u}(t)}\frac{\partial \hat{u}(t)}{\partial \theta_{ij}} = -e(t)\cdot y_{\hat{u}}(t)\cdot\frac{\partial \hat{u}(t)}{\partial a}\frac{\partial a}{\partial \theta_{ij}} = -e(t)\cdot y_{\hat{u}}(t)\left(\frac{w_{kl}\cdot c - b}{c^2}\right)a\left(\frac{2(x_i - \theta_{ij})}{\sigma^2}\right) = -e(t)\cdot y_{\hat{u}}(t)\left(\frac{w_{kl} - \hat{u}}{c}\right)\left(\frac{2a(x_i - \theta_{ij})}{\sigma^2}\right), \tag{3.13}$$

$$\frac{\partial J_2}{\partial \sigma_{ij}} = e(t)\frac{\partial e(t)}{\partial \sigma_{ij}} = -e(t)\frac{\partial y(t)}{\partial \sigma_{ij}} =$$

$$-e(t)\frac{\partial y(t)}{\partial \hat{u}(t)}\frac{\partial \hat{u}(t)}{\partial \sigma_{ij}} = -e(t)\cdot y_{\hat{u}}(t)\cdot\frac{\partial \hat{u}(t)}{\partial a}\frac{\partial a}{\partial \sigma_{ij}} =$$

$$-e(t)\cdot y_{\hat{u}}(t)\left(\frac{w_{kl}\cdot c - b}{c^2}\right)a\left(\frac{2(x_i-\theta_{ij})}{\sigma^3}\right) =$$

$$-e(t)\cdot y_{\hat{u}}(t)\left(\frac{w_{kl}-\hat{u}}{c}\right)\left(\frac{2a(x_i-\theta_{ij})^2}{\sigma^3}\right), \quad (3.14)$$

where

$$y_{\hat{u}}(t) = \frac{\partial y(t)}{\partial \hat{u}(t)} = \frac{y(t+1)-y(t)}{\hat{u}(t+1)-\hat{u}(t)}. \quad (3.15)$$

The following equation shows the parameters adjustment by means of gradient descent algorithm

$$w_{kl}(t+1) = w_{kl}(t)+\Delta w_{kl}(t) = w_{kl}(t)-\eta_w\left[\frac{\partial J_2}{\partial w_{kl}}\right], \quad (3.16)$$

$$\theta_{ij}(t+1) = \theta_{ij}(t)+\Delta\theta_{ij}(t) = \theta_{ij}(t)-\eta_\theta\left[\frac{\partial J_2}{\partial \theta_{ij}}\right], \quad (3.17)$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t)+\Delta\sigma_{ij}(t) = \sigma_{ij}(t)-\eta_\sigma\left[\frac{\partial J_2}{\partial \sigma_{ij}}\right], \quad (3.18)$$

where η_w, η_θ, η_σ represent the learning rate of w, θ, and σ respectively, and $0 < \eta_w, \eta_\theta, \eta_\sigma < 1$.

The gradient descent algorithm is as follows:

1) θ^{**}, σ^{**} and w^{**} are approximate global optimal parameters from chaos searching, they are substituted in Eq. (3.11) as the initial values of gradient descent searching.

2) According to Eqs. (3.12) ~ (3.15), gradient is computed and substituted in Eqs. (3.16) ~ (3.18) to adjust global approximate optimal values θ^{**}, σ^{**} and w^{**}.

3) Use Eq. (3.10) to calculate performance index J_2, if it coincides with the requirement condition, the iteration stops, otherwise, let $\theta^{**} = \hat{\theta}^{**}$, $\sigma^{**} = \hat{\sigma}^{**}$, $w^{**} = \hat{w}^{**}$ be the initial values for substituting in Eq. (3.11) and Step 2) is repeated.

4 Simulation study

In system simulation, let $T(x_1) = T(x_2) = 7$, therefore, the FNN is composed of the 2-14-49-1 neuron. Controlled plant is a second order system with time-delay of the following form

$$G(s) = \frac{Ke^{-\tau s}}{T_1 s^2 + T_2 s + 1}. \quad (4.1)$$

Initial values of each chaos variable are sampled at random in the (0,1) interval, $J_1^* = 10^{-2}$, $\lambda = 0.25$, $\alpha = 0.05$, $N = 10^3$. Parameters of the plant are $K = 5$, $T_1 = 4$, $T_2 = 5$, $\tau = 0.2$. After off-line optimizing with chaos algorithm, the redundant nodes are deleted, the link weights w_{kl}^{**} of Layer Ⅳ are listed in Table 1, "—" denotes that correspondent node is nonexistent, the FNN is composed of the 2 - 14 - 40 - 1. The center parameters θ_{ij}^{**} of Layer Ⅱ are listed in Table 2 and width parameters σ_{ij}^{**} are listed in Table 3.

Table 1 Parameters of weight w_{kl}^{**}

ec \ e	NL	NM	NS	Z	PS	PM	PL
NL	0.7870	-0.1646	0.5774	—	-0.9440	0.8621	-0.6476
NM	0.1653	-0.2820	—	-0.6498	—	0.6098	-0.6259
NS	-0.4107	-0.3713	0.7776	-0.1979	-0.3512	0.5183	-0.3669
Z	0.1788	—	-0.1712	—	-0.1792	—	0.7750
PS	-0.1460	0.5921	-0.1054	—	0.0858	0.3013	0.7096
PM	-0.7935	—	-0.2247	-0.1019	0.5809	0.3877	0.5590
PL	0.6792	0.7996	0.6447	—	-0.6505	-0.7864	-0.9927

Table 2 Parameters of the center θ_{ij}^{**}

e	-0.9950	-0.7845	-0.4135	0.0838	0.2452	0.5233	0.8756
ec	-0.8722	-0.6975	-0.2274	0.1395	0.4911	0.7413	0.9979

Table 3 Parameters of the width σ_{ij}^{**}

e	0.0077	0.1007	0.0062	0.1102	0.0055	0.1003	0.1932
ec	0.2632	0.2648	0.0300	0.0007	0.1059	0.0848	0.0381

The FNN is connected to the system after chaos optimization, the above parameters θ_{ij}^{**}, σ_{ij}^{**} and w_{kl}^{**} are the initial values for gradient descent on-line searching. We pick parameters $\eta_w = 0.1$, and $\eta_\sigma = 0.001, \eta_\theta = 0.001, J_2^* = 10^{-3}$. After on-line optimizing with gradient descent algorithm, the system can find out the global optimal values. Compare the optimal method of the paper with method of rule self-adjusting fuzzy controller in literature [6], the simulation curves of system output is shown in Fig.2. When random disturbance are added to the system, peak value is 0.1, the output curve of system after chaos and gradient descent optimization is shown in Fig.3. Change the parameters of the controlled plant and keep other conditions unchanged, the output curves of the system are shown in Fig.4.

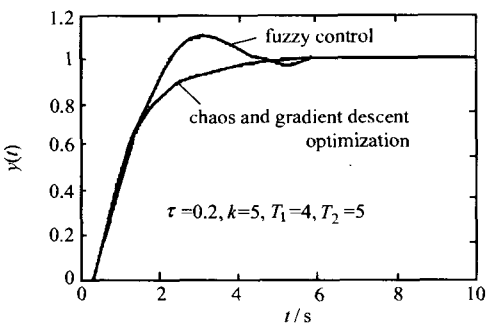


Fig. 2 Output curve of second order delay system

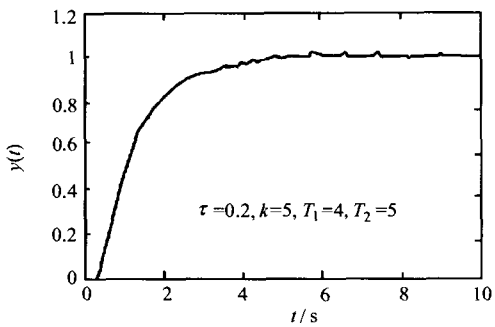
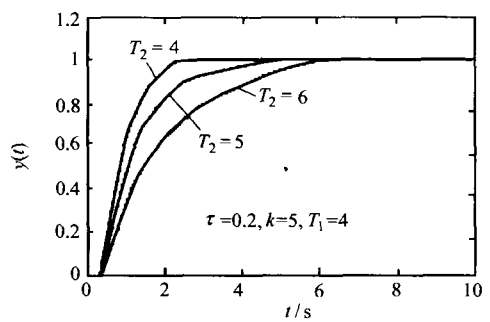


Fig. 3 Output curve under interference



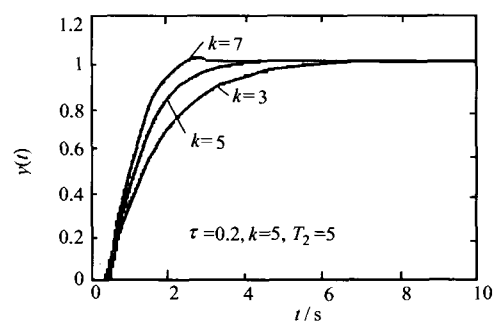


Fig. 4 Robustness test for second order system with time-delay

5 Conclusion

A new hybrid optimal scheme combining chaos algorithm and gradient descent algorithm has been proposed in this paper. The new scheme combines the advantages of them, and improves the efficiency of the chaos searching process and overcomes the gradient descent shortcomings of slow convergence and of easy to be trapped in local minimum. The design of FNN has the features of global optimization. Simulation results show that the hybrid optimal scheme is superior to the method in [6], and this optimization method is efficient in searching for controller parameters.

References:

[1] LIN C T, GEORGE C S L. Neural-network-based fuzzy logic control and decision system [J]. *IEEE Trans on Computers*, 1991, 40(12): 1320 - 1336.

[2] ISHIBUCHI H, KWON K, TANAKA H. A learning algorithm of fuzzy neural networks with triangular fuzzy weights [J]. *Fuzzy Sets and Systems*, 1995, (71): 277 - 293.

[3] LUONAN C, KAZUYUKI A. Chaotic simulated annealing by a neural network model with transient chaos [J]. *Neural Networks*, 1995, 8(6): 915 - 930.

[4] ISAO T, KAZUYUKI A, TOMOMASA N. Adaptive annealing for chaotic optimization[J]. *Physical Review E*, 1998, 58(4): 5157 - 5160.

[5] HINTON G E, NOWLAN S J. How learning can guide evolution [J]. *Complex Systems*, 1987, 1(3): 495 - 502.

[6] HE S Z, TAN S, WANG P Z. Fuzzy self-tuning of PID controllers [J]. *Fuzzy Sets and Systems*, 1993, (56): 37 - 46.

作者简介:

邹 恩 (1956—),女,教授,毕业于中南大学信息科学与工程学院并获硕士学位,现在该校攻读博士学位,发表论文20多篇,研究方向为神经网络、模糊控制,混沌理论与混沌优化等, E-mail: zouen10@sina.com;

李祥飞 (1969—),男,副教授,2003年毕业于中南大学并获博士学位,发表论文10多篇,研究方向为神经网络、混沌优化、最优控制;

张泰山 (1940—),教授,博士生导师,发表论文50余篇,研究方向为模糊控制、神经网络、混沌控制与优化、遗传算法、进化计算.