Int. J. Mach. Learn. & Cyber. (2014) 5:815–822
DOI 10.1007/s13042-013-0224-y

ORIGINAL ARTICLE

Parameters optimization of polygonal fuzzy neural networks based on GA-BP hybrid algorithm

Yongqiang Yang · Guijun Wang · Yang Yang

Received: 16 September 2013 / Accepted: 15 December 2013 / Published online: 7 January 2014

Abstract Based on the extensive operations of polygonal fuzzy numbers, a GA-BP hybrid algorithm for polygonal fuzzy neural network is designed. Firstly, an optimal solution is obtained by the global searching ability of GA algorithm for the untrained polygonal fuzzy neural network. Secondly, some parameters for connection weights and threshold values are appropriately optimized by using an improved BP algorithm. Finally, through a simulation example, we demonstrate that the GA-BP hybrid algorithm based on the polygonal fuzzy neural network can not only avoid the initial values' dependence and local convergence of the original BP algorithm, but also overcome a blindness problem of the traditional GA algorithm.

Keywords Polygonal fuzzy numbers · Polygonal fuzzy neural network · GA-BP hybrid algorithm · Parameters optimization

1 Introduction

In recent years, it is an increasing interest that is combing artificial neural networks with fuzzy system. Fuzzy neural networks (FNN), which is made up of artificial neural networks and fuzzy system, can learn, identify and process fuzzy information. So FNN is one of the most successful examples to bring the artificial networks and fuzzy system together [1, 2]. But FNN based on Zadeh's extension principle has not satisfied linearity so that some extensive applications of FNN can be greatly restricted. Consequently, in order to realize a nonlinear operation between fuzzy numbers and improve the ability of universal approximation of fuzzy neural networks, polygonal FNN is introduced. The polygonal FNN is a new network which depends on polygonal fuzzy numbers and artificial neural networks. The connection weights and threshold of polygonal FNN both take values of polygonal fuzzy numbers. Furthermore, the topology of polygonal FNN is intuitive and clear, and operations of polygonal FNN are simple and satisfy the linearity. So, the polygonal FNN has a stronger approximation ability than fuzzy neural network based on Zadeh's extensive principle [3, 4]. Generally, when the network is fixed, its performance is determined by parameters of the network. However, parameters optimization of polygonal FNNs mainly embody in networks' learning algorithms. Therefore, how to optimize parameters of polygonal FNNs is an important goal of the networks' learning algorithms.

In 2002, Liu [3] first developed a concept of n-symmetric polygonal fuzzy numbers and established a polygonal fuzzy neural network. Because the polygonal FNNs rely on the linear operations of polygonal fuzzy numbers to design a learning algorithm, learning algorithms of the polygonal FNNs are more convenient to compute than those of the FNNs. Next, according to the above characteristic, through presenting a traditional Back Propagation (BP) algorithm for polygonal FNNs to realize parameters optimization, Liu [4] verified the polygonal FNNs had a better performance in the application than FNNs. But initial values' dependence and local convergence of the

This work has been supported by National Natural Science Foundation China (Grant No. 61374009)

Y. Yang · G. Wang (✉) · Y. Yang
School of Mathematics Sciences, Tianjin Normal University, Tianjin 300387, China
e-mail: tjwgj@126.com

Y. Yang
e-mail: 510176215@qq.com

traditional BP algorithm has not been resolved. Thus, it is necessary to design some new and effective learning algorithms for polygonal FNNs. In 2011, a traditional genetic algorithm (GA), which is based on natural selection and genetic principles, was used for optimizing parameters of polygonal FNNs [5]. However, the traditional GA which is a kind of global searching algorithm has a blindness problem to make the optimization ineffective. Besides, recently, although some learning algorithms were proposed to optimize the polygonal FNNs [6–9], there were many aspects to be improved. In fact, through analyzing feature of the BP and GA algorithms, we can find that the global searching capability of GA can resolve the traditional BP algorithm' deficiencies. And, feasibility and local searching ability of BP algorithm can make the traditional GA get rid of blindness problem. Thus, now, we present parameters optimization of polygonal FNNs based on GA-BP hybrid algorithm.

In this paper, we introduce the polygonal FNN based on the extensive operations of polygonal fuzzy numbers. And then, according to their extensive operations, we improve the traditional BP [3, 4] and GA algorithm [5] based on the polygonal FNN. At the same time, using the improved GA to search a global optimal solution for the untrained polygonal FNN, and combining the improved BP algorithm to optimize the connection weights and threshold values, we design a GA-BP hybrid algorithm for polygonal FNN. Finally, through a simulation example, the results show that a GA-BP hybrid algorithm doesn't only overcome the initial values' dependence and local convergence of the BP algorithms and blindness problem of the traditional GA, but also improves the convergence rate of the polygonal FNN.

In addition, the paper is organized as follows: After the introduction, some basic notations for polygonal fuzzy numbers and its graphical explanation are briefly summarized in Sect. 2. In Sect. 3, the structural model and the error function of the polygonal fuzzy neural network are given. Section 4 shows that a new GA-BP hybrid algorithm and its parameters optimization are designed. In Sect. 5, a simulation example for the polygonal FNN is realized. Some conclusions are indicated in the final section.

2 Polygonal fuzzy numbers

It is well known that one of the simplest fuzzy numbers, even the triangular fuzzy numbers, doesn't satisfy linear operation, so the application of fuzzy numbers is a difficult problem. This also raises an important issue that how to approximately achieve a nonlinear operation between fuzzy numbers. And how to construct the appropriate FNN to approximate a given nonlinear function can be very significant. For this, Liu [3] was the first to put forward the concept of n-symmetric polygonal fuzzy number, and its operations can be linearized. But this polygonal fuzzy numbers are a little shortage, and he [6] revised and improved it.

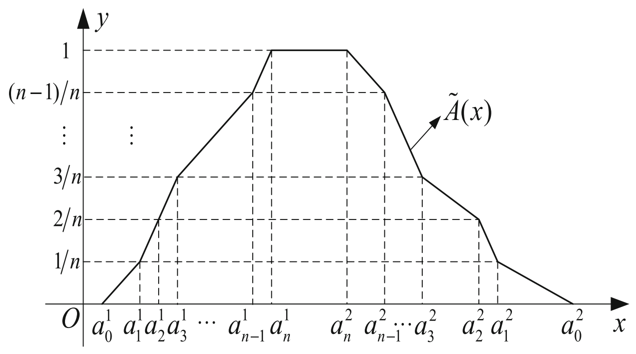
Throughout this paper, we always let $\mathbb{R}$denote the set of all real numbers, $\mathbb{R}^+$the set of non-negative real numbers and $\mathbb{N}$ the family of all natural numbers. Let sign $\|\cdot\|$ be a Euclid norm, $F_0(\mathbb{R})$ the family of all of fuzzy numbers on $\mathbb{R}$.

Definition 1 [3, 6] Let $\tilde{A} \in F_0(\mathbb{R})$,for given $n \in \mathbb{N}$, divide the closed interval $[0, 1]$ along the y-axis into n equal-sized closed intervals bounded by points $x_i = i/n$, $i = 1, 2, \cdots, n-1$. If there exists a set of ordered real numbers: $a_0^1, a_1^1, \cdots, a_n^1, a_n^2, \cdots, a_1^2, a_0^2 \in \mathbb{R}$ with $a_0^1 \le a_1^1 \le \cdots \le a_n^1 \le a_n^2 \le \cdots \le a_1^2 \le a_0^2$ such that $\tilde{A}(a_i^q) = i/n$, where $q = 1, 2$ and membership function $\tilde{A}(x)$ takes straight lines in $[a_{i-1}^1,\ a_i^1]$ and $[a_i^2,\ a_{i-1}^2]$, where $i = 1, 2, \ldots, n$ (see Figs. 1, 2).

Then $\tilde{A}$ is called an $n-$polygonal fuzzy number, for simplicity, it is always denoted as follows

$\tilde{A} = (a_0^1, a_1^1, \cdots, a_n^1, a_n^2, \cdots, a_1^2, a_0^2).$

Obviously, an $n-$polygonal fuzzy number $\tilde{A}$ can be completely determined by the finite ordered points $a_0^1, a_1^1, \cdots, a_n^1, a_n^2, \cdots, a_1^2, a_0^2$ on $x-$axis, where the coordinate of each vertex is$(a_i^q,\ i/n)$, $q = 1, 2$; $i = 0, 1, \cdots, n$.



y
1
(n−1)/n
3/n
2/n
1/n
O
$\tilde{A}(x)$
a_0^1 $a_1^1 a_2^1 a_3^1$ ⋯ $a_{n-1}^1 a_n^1$ a_n^2 $a_{n-1}^2 \cdots a_3^2$ $a_2^2 a_1^2$ a_0^2
x

Fig. 1 Image of $n-$ polygonal fuzzy number $\tilde{A}$

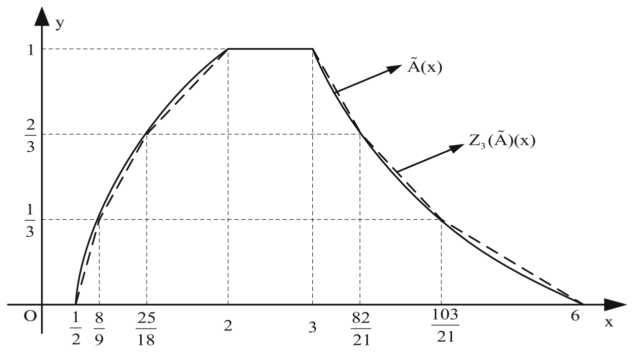


Fig. 2 Mixed image of $\tilde{A}$ and its 2- polygonal fuzzy number

Generally, the value of n determines the size of nodes on the polyline. Specially, if $n=1$, a $1-$polygonal fuzzy number reduces to a trapezoidal or triangle fuzzy number. For any given fuzzy number and $n \in \mathbb{N}$, we can get only one $n-$polygonal fuzzy number by method in Fig. 1, to replace the original fuzzy number (see Fig. 2).

Let $F_0^{tn}(\mathbb{R})$ denote the set of all $n-$ polygonal fuzzy numbers on $\mathbb{R}$, and $\tilde{A} \in F_0^{tn}(\mathbb{R}^+) \Leftrightarrow \tilde{A}(x)=0,\ \forall x<0$. In addition, extended linear operators of polygonal fuzzy numbers can be described in papers [6–8], This article will do not repeat.

3 Polygonal FNN

Polygonal FNN is a kind of new network system, its connection weights and threshold are polygonal fuzzy numbers, so its internal processing are based on the extension operations of polygonal fuzzy numbers. In fact, the network's expression is a system that contains the arithmetics of polygonal fuzzy numbers, and the network can process information by a finite number of real numbers. The research object of this article is a single input single output (for short SISO) three-layer forward polygonal FNN.

If the input variable $\tilde{X}$, the connection weights $\tilde{U}_j$ and $\tilde{V}_j$, the threshold $\tilde{\Theta}_j$ of the hidden layer are taken values in $F_0^{tn}(\mathbb{R})$, and let an activation function σ be a continuous monotonic Sigmoidal function, then SISO three-layer forward polygonal FNN can be denoted as follow

$$\tilde{Y}=\tilde{F}_{nn}(\tilde{X})=\sum\nolimits_{j=1}^{p}\tilde{V}_j\cdot\sigma(\tilde{U}_j\cdot\tilde{X}+\tilde{\Theta}_j).$$

For given untrained polygonal fuzzy pattern pairs $(\tilde{X}(1),\tilde{O}(1)),(\tilde{X}(2),\tilde{O}(2)),\ldots,(\tilde{X}(L),\tilde{O}(L))$, where $\tilde{X}(l)$, $\tilde{O}(l)\in F_0^{tn}(\mathbb{R}^+)$, and $\tilde{X}(l)$ denote network's input, $\tilde{O}(l)$ denote network's expected output. If the network's actual output can be denoted as $\tilde{Y}(l)$, then $\tilde{Y}(l)=\tilde{F}_{nn}(\tilde{X}(l))$, $l=1,2,\cdots,L$. This moment, let

$$\tilde{X}(l)=(x_0^1(l),x_1^1(l),\cdots,x_n^1(l),x_n^2(l),\cdots,x_1^2(l),x_0^2(l)),$$
$$\tilde{Y}(l)=(y_0^1(l),y_1^1(l),\cdots,y_n^1(l),y_n^2(l),\cdots,y_1^2(l),y_0^2(l)),$$
$$\tilde{O}(l)=(o_0^1(l),o_1^1(l),\cdots,o_n^1(l),o_n^2(l),\cdots,o_1^2(l),o_0^2(l)).$$

According to the metric D of polygonal fuzzy number [3], we may define the error function E of polygonal FNN as follows

$$\begin{aligned}E&=\frac{1}{2}\sum_{l=1}^{L}D(\tilde{O}(l),\tilde{Y}(l))^2\\&=\frac{1}{2}\sum_{l=1}^{L}\left(\sum_{i=0}^{n}((o_i^1(l)-y_i^1(l))^2+(o_i^2(l)-y_i^2(l))^2)\right).\end{aligned}\qquad(1)$$

In terms of input variable $\tilde{X}$, the network can gradually optimized connection weights $\tilde{U}_j$, $\tilde{V}_j$ and threshold $\tilde{\Theta}_j$ by learning. And this optimization can make $\tilde{O}(l)$ approximate or equal $\tilde{Y}(l)$. In addition, we can also put all of adjustable parameters $u_i^q(j)$, $v_i^q(j)$, $\theta_i^q(j)$ $(i=0,1,\cdots,n;\ j=1,2,\cdots,p;\ q=1,2)$ together to express a form of parameters vector, denoted as w, that is

$$\begin{aligned}w&=(w_1,w_2,\cdots w_N)=(u_0^1(1),\cdots,u_0^2(1),\cdots,\\&u_0^1(p),\cdots,u_0^2(p),v_0^1(1),\cdots,v_0^2(1),\cdots,v_0^1(p),\cdots,\\&v_0^2(p),\theta_0^1(1),\cdots,\theta_0^2(1),\cdots,\theta_0^1(p),\cdots,\theta_0^2(p))\end{aligned}\qquad(2)$$

Aim at (1), the error function E can be denoted as $E(w)$, and its gradient vector can be denoted as

$$\nabla E(w\,[t])=(\partial E(w)/\partial w_1,\partial E(w)/\partial w_2,\cdots,\partial E(w)/\partial w_N).$$

Lemma 1 [4] *Let $E(w)$ denote the error function defined by formula (1), then for $i=0,1,\cdots,n;\ j=1,2,\cdots,p;q=1,2$, $E(w)$ is almost everywhere differentiable in $\mathbb{R}^N$, and its partial derivate formula $\partial E/\partial u_i^q(j)$, $\partial E/\partial v_i^q(j)$, $\partial E/\partial\theta_i^q(j)$ can exist. (Detailed expressions of several partial derivative can be found.(see [4]))*

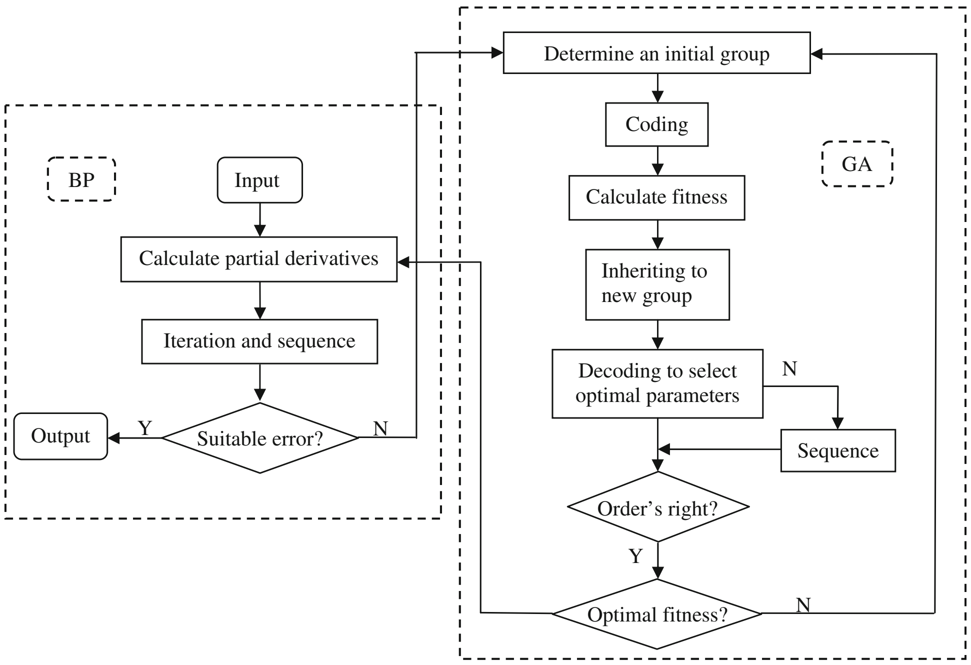
4 GA-BP hybrid algorithm

An original BP algorithm is one of the common learning algorithms in neural networks, and it is not only simple and easy to implement, but also has a strong local search ability. However, BP algorithm has slower convergence speed and is easy to fall into minimum. For example, when there are many extreme points in solution space, and the initial parameters are not appropriate, BP algorithm can be easy to fall into local convergence point to result in a bad situation surely. In addition, a GA algorithm is a global optimization algorithm. It combines natural selection in biology with population evolution. The GA algorithm can code parameters into a space of the individual, and it also can select the optimal individual in groups by genetic manipulation, then decode the optimal individual to restore parameters to achieve optimization. But a traditional GA algorithm's iterative process is based on the same original group, so it maybe can't select the optimal gene in the original group. Hence, the traditional GA algorithm can search for the sub-optimal solution, sometimes it happens "inbreeding" phenomenon (see [10–12]).

Accordingly, this paper combines the advantages of BP and GA algorithm, and proposes a GA-BP hybrid algorithm based on parameters optimization for polygonal FNN (see Fig. 3).

From Fig. 3, with GA and BP interacting each other, we can see that many parameters can be optimized by BP

Fig. 3 Schematic diagram of GA-BP hybrid algorithm



algorithm (see [13, 14]). In addition, according to gradient vector $\nabla E(w)$ from the partial derivative formula of Lemma 1, we can design the GA-BP hybrid algorithm based on polygonal FNN as follows.

Algorithm 1 (BP algorithm part): Based on polygonal FNN, we improve the original BP algorithm to make arithmetics of its neurons meet extensive operations of polygonal fuzzy numbers. At the same time, we can introduce dynamic learning constant and momentum constant to accelerate searching rate. Then, parameters of the network can be optimized to design the following algorithm.

Step 1. For $j = 1, 2, \cdots, p$, let connection weights $\tilde{U}_j[t_1]$, $\tilde{V}_j[t_1]$ and threshold $\tilde{\Theta}_j[t_1] \in F_0^{tn}(\mathbb{R}^+)$ denote

$$\tilde{U}_j[t_1] = (u_0^1(j)[t_1], u_1^1(j)[t_1], \cdots, u_n^1(j)[t_1], u_n^2(j)[t_1], \cdots, u_1^2(j)[t_1], u_0^2(j)[t_1]),$$

$$\tilde{V}_j[t_1] = (v_0^1(j)[t_1], v_1^1(j)[t_1], \cdots, v_n^1(j)[t_1], v_n^2(j)[t_1], \cdots, v_1^2(j)[t_1], v_0^2(j)[t_1]),$$

$$\tilde{\Theta}_j[t_1] = (\theta_0^1(j)[t_1], \theta_1^1(j)[t_1], \cdots, \theta_n^1(j)[t_1], \theta_n^2(j)[t_1], \cdots, \theta_1^2(j)[t_1], \theta_0^2(j)[t_1]).$$

We make the number of iteration step $t_1 = 0$, and randomly select initial values $\tilde{U}_j[0], \tilde{V}_j[0], \tilde{\Theta}_j[0]$ $(j = 1, 2, \cdots, p)$, and for given error accuracy $\varepsilon > 0$.

Step 2. For $l = 1, 2, \cdots, L; j = 1, 2, \cdots, p; i = 0, 1, \cdots, n; q = 1, 2$, according to Lemma 1, we calculate the partial derivatives $\partial E/\partial u_i^q(j), \partial E/\partial v_i^q(j), \partial E/\partial \theta_i^q(j)$, where the iterative formula of the connection weights' component $u_i^q(j)$, $v_i^q(j)$ and threshold's component $\theta_i^q(j)$ can be denoted as

$$\begin{cases} u_i^q(j)[t+1] = u_i^q(j)[t] + \eta \cdot \dfrac{\partial E}{\partial u_i^q(j)[t]} + \alpha \cdot \Delta u_i^q(j)[t]; \\ v_i^q(j)[t+1] = v_i^q(j)[t] + \eta \cdot \dfrac{\partial E}{\partial v_i^q(j)[t]} + \alpha \cdot \Delta v_i^q(j)[t]; \\ \theta_i^q(j)[t+1] = \theta_i^q(j)[t] + \eta \cdot \dfrac{\partial E}{\partial \theta_i^q(j)[t]} + \alpha \cdot \Delta \theta_i^q(j)[t]. \end{cases}$$

where differentials $\Delta u_i^q(j)[t_1] = u_i^q(j)[t_1] - u_i^q(j)[t_1 - 1]$, $\Delta v_i^q(j)[t_1] = v_i^q(j)[t_1] - v_i^q(j)[t_1 - 1]$, $\Delta \theta_i^q(j)[t_1] = \theta_i^q(j)[t_1] - \theta_i^q(j)[t_1 - 1]$, this η, α denote learning constant and momentum constant.

Step 3. Through the above iteration formulas, we can calculate components $u_i^q(j)[t_1+1], v_i^q(j)[t_1+1], \theta_i^q(j)[t_1+1]$ $(q = 1, 2; i = 0, 1, \cdots, n; j = 1, 2, \cdots, p)$. According to incrementally reorder, the new order relation of polygonal fuzzy numbers are obtained as follows:

$$u_1'^0(j)[t_1+1] \le u_1'^1(j)[t_1+1] \le \cdots \le u_n'^1(j)[t_1+1] \le u_n'^2(j)[t_1+1] \le \cdots \le u_1'^2(j)[t_1+1] \le u_0'^2(j)[t_1+1];$$

$$v_0'^1(j)[t_1+1] \le v_1'^1(j)[t_1+1] \le \cdots \le v_n'^1(j)[t_1+1] \le v_n'^2(j)[t_1+1] \le \cdots \le v_1'^2(j)[t_1+1] \le v_0'^2(j)[t_1+1];$$

$$\theta_0'^1(j)[t_1+1] \le \theta_1'^1(j)[t_1+1] \le \cdots \le \theta_n'^1(j)[t_1+1] \le \theta_n'^2(j)[t_1+1] \le \cdots \le \theta_1'^2(j)[t_1+1] \le \theta_0'^2(j)[t_1+1].$$

Step 4. For $\varepsilon > 0$ and $l = 1, 2, \cdots, L$, whether $D(\tilde{Y}(l), \tilde{O}(l)) < \varepsilon$. If they fulfill, then output $\tilde{U}_j[t_1]$, $\tilde{V}_j[t_1]$, $\tilde{\Theta}_j[t_1]$ $(j = 1, 2, \cdots, p)$ and $\tilde{O}(l)$ $(l = 1, 2, \cdots, L)$; otherwise, let $t_1 = t_1 + 1$, and turn to Algorithm 2.

Note 1. According to [15–17], based on extensive operations of polygonal fuzzy numbers, we can improve learning constant η and momentum constant α in Step 3 to make them become functions about iteration step t_1. Let

$$\eta = \eta[t_1] = \frac{\rho_0 \cdot E(w\,[t_1])}{\|\nabla E(w\,[t_1])\|^2};$$
$$\alpha = \alpha[t_1] = \frac{|\Delta E(w\,[t_1])|}{\sum_{k=1}^{t_1-1} |\Delta E(w\,[k])|},$$

where, $\Delta E(w\,[t_1]) = E(w\,[t_1]) - E(w\,[t_1 - 1])$, $w\,[t_1]$ denote the parameter's vector of t_1 step. Clearly, when $t_1 = 0$, $E(w\,[0])$ is usually a large number, i.e. $E(w\,[0]) = 100$. And $\rho_0$is a small constant, i.e. 0.01.$\nabla E(w\,[t_1])$ can be denoted as gradient vector of E in step t_1, so we can get that

$$\nabla E(w[t_1]) = (\partial E(w)/\partial w_1, \partial E(w)/\partial w_2, \ldots, \partial E(w)/\partial w_N).$$

Algorithm 2 (GA algorithm part): Based on polygonal FNN, we can use network's parameters as an initial group of GA algorithm. The original single group can be denoted as multiple groups, and we can also adjust crossover and mutation probability dynamically. This improved GA algorithm is not only easy to combine with BP algorithm, but also avoids the above disadvantages. And it can search for the global optimal solution to optimize parameters better than the traditional GA algorithm.

Step 1. Given searching accuracy $\gamma > 0$, and the number of iteration step $t_2 = 0$, for $j = 1, 2, \cdots, p$, let

$$\begin{cases} a_j[t_2] = \min\limits_{0 \le i \le n;\, q=1,2} u_i^q(j)[t_1] \\ a_{j+p}[t_2] = \min\limits_{0 \le i \le n;\, q=1,2} v_i^q(j)[t_1] \\ a_{j+2p}[t_2] = \min\limits_{0 \le i \le n;\, q=1,2} \theta_i^q(j)[t_1] \end{cases};$$
$$\begin{cases} b_j[t_2] = \max\limits_{0 \le i \le n;\, q=1,2} u_i^q(j)[t_1] \\ b_{j+p}[t_2] = \max\limits_{0 \le i \le n;\, q=1,2} v_i^q(j)[t_1] \\ b_{j+2p}[t_2] = \max\limits_{0 \le i \le n;\, q=1,2} \theta_i^q(j)[t_1] \end{cases},$$

Step 2. For each$j = 1, 2, \cdots, 3p$, let each parameter's vector of binary string length denote $l_j = \log_2((a_j[t_2] - b_j[t_2])/\gamma)$ In addition, l denotes a group's gene pool of binary string length, i.e. $l = \max\{l_1, l_2, \ldots, l_{3p}\}$. According to (2), we let $w\,[t_2] = (w_1[t_2], w_2[t_2], \ldots, w_N[t_2])$, and for all $i = 1, 2, \ldots, N$, and approximately, each parameter's component can be expressed as a binary number $\beta_i[t_2] \in \varpi_{nn}(w_i[t_2] \to \beta_i[t_2])$.

Step 3. Based on Algorithm 1, we randomly generate m_0 groups of N individuals. (Total of groups expands as m_0.) To calculate easily, we introduce fitness function $J(w) = 1/(1 + E(w))$, and for $\beta_i(k)[t_2] (i = 1, 2, \ldots, N;$ $k = 1, 2, \ldots, m_0)$, decode to transform $w'(k)\ [t_2] = (w_1'(k)$ $[t_2], w_2'(k)[t_2], \ldots, w_N'(k)[t_2])$, then calculate $J(w'(k)[t_2])$.

Step 4. According to the roulette wheel selection (see [9]), for $k = 1, 2, \ldots, m_0$, let survival probability of each individual $\beta_i(k)[t_2]$ denote $p_k = J(w'(k)[t_2]) / \sum_{k'=1}^{m_0} J(w'(k')[t_2])$. Let single-point crossover operator $p : \varpi_{nn}^2 \to \varpi_{nn}^2$, randomly select two-to-two individuals $(\beta_i(k_1)[t_2], \beta_i(k_2)[t_2])$ from m_0 groups, then based on crossover probability $p(\beta_i(k_1)[t_2], \beta_i(k_2)[t_2])$, we can get two new individuals $(\beta_i(k_1')[t_2 + 1], \beta_i(k_2')[t_2 + 1])$. In addition, the group of new individuals can be denoted as $\beta_i'(k)[t_2 + 1](i = 1, 2, \ldots, N; k = 1, 2, \ldots, m_0)$.

Step 5. For all $i = 1, 2, \ldots, N; k = 1, 2, \ldots, m_0$, decode component $\beta_i'(k)[t_2]$, then transform $w'(k)[t_2 + 1] = (w_1'(k)[t_2 + 1],\ w_2'(k)[t_2 + 1], \ldots, w_N'(k)[t_2 + 1])$, so we need to recalculate $J(w'(k)[t_2 + 1])$, and let

$$w_{best}[t_2 + 1] = \max\{J(w'(k)[t_2 + 1]) \mid k = 1, 2, \ldots, m_0\}.$$

Step 6. Judge $k = k_0$, for each $i = 0, 1, 2,\ \ldots, n$; $j = 1, 2, \ldots, p$, whether $w_{best}[t_2 + 1]$ satisfies the following formula.

$$w_i'^1(j)(k_0)[t_2 + 1] \le w_{i+1}'^1(j)(k_0)[t_2 + 1] \le w_{i+1}'^2(j)(k_0)[t_2 + 1] \le w_i'^2(j)(k_0)[t_2 + 1].$$

If the above expression is established, then turn to step 7; otherwise, for all $j = 1, 2, \ldots, p$, sort with selection method from small to big, and turn to step 7.

Step 7. Judge whether $J(w'(k_0)[t_2 + 1]) \ge J(w'(k_0)[t_2])$. If they fulfill, let $t_1 = t_1 + 1$, and turn to step 2 in Algorithm 1; otherwise, let $t_2 = t_2 + 1$, and turn to step 1 in Algorithm 2.

Note 2. The problem in this paper is a discrete optimization problem, so in the section of GA algorithm, we can use the binary to represent each individual's gene. This method can make encoding and decoding operations easy, and simplify crossover and mutation evolutionary processes.

Note 3. In Algorithm 2, to maintain the diversity of population and parameters optimization, and prevent premature phenomenon, we can use dynamic crossover and mutation probability. Let the number of evolution step λ= 100, for $k = 1, 2, \ldots,\ m_0$, and crossover probability denotes $\xi_k[t_2]$, mutation probability denotes $\zeta_k[t_2]$, then their formulas are that

$$\xi_k[t_2] = \begin{cases} \frac{\xi_{\max}}{1 + t_2/\lambda}, & \xi_k[t_2] > \xi_{\min} \\ \xi_{\min}, & \xi_k[t_2] \le \xi_{\min} \end{cases};$$
$$\zeta_k[t_2] = \begin{cases} \zeta_{\max} \cdot e^{-\mu t_2/\lambda}, & \zeta_k[t_2] > \zeta_{\min} \\ \zeta_{\min}, & \zeta_k[t_2] \le \zeta_{\min} \end{cases}.$$

Generally, crossover probability $\xi_k[t_2]$ is taken values in 0.4–0.99, and mutation probability $\zeta_k[t_2]$ is taken values in 0.001–0.1. So we can let minimum of crossover probability $\xi_{\min} = 0.4$, maximum of crossover probability $\xi_{\max} = 0.99$, minimum of mutation probability $\zeta_{\min} = 0.001$, and maximum of mutation probability $\zeta_{\max} = 0.1$, and $\mu = 0.1$.

Note 4.The decoding process in step 3 and step 5 of Algorithm 2 is the following process. Firstly, we can calculate the decimal number $d_i(j)$ based on the corresponding

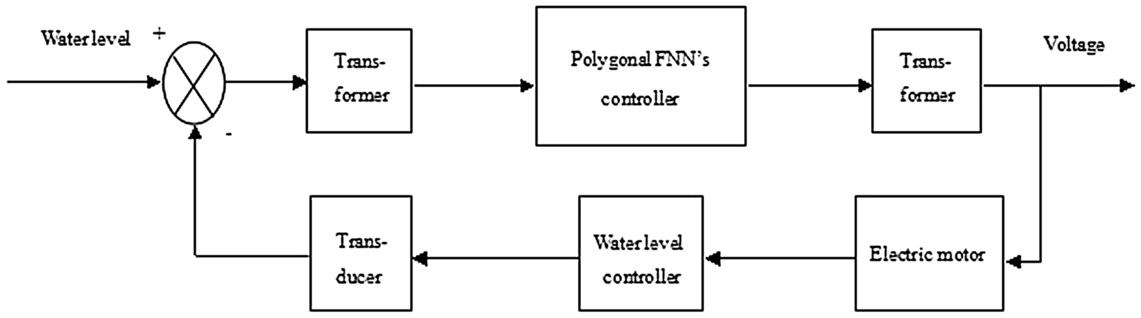


Fig. 4 Structure of boiler drums water level control system

binary number. Then, according to the following formula, we can get the actual network's parameters $w_i(j)[t_2]$: $w_i(j)[t_2] = d_i(j) \cdot \gamma' + a_j[t_2]\ (i = 1, 2, \ldots, 2n + 2; j = 1, 2, \ldots, p)$, where γ' denotes actual searching accuracy after encoding, and the computing formula of γ' is that $\gamma' = (b_j[t_2] - a_j[t_2])/(2^l - 1)$.

5 A simulation example

To discuss parameters optimization's influence of polygonal FNN based on a GA-BP hybrid algorithm, this paper can apply the optimization to the simulation model of boiler drum water level control (see [18, 19]). We can see the structure of model system as follows (see Fig. 4).

Generally, boiler drum water level control model can be expressed as $G(s) = k/s(Ts + 1)$, where s denotes level, k denotes rate of water level, and T denotes time (see [20, 21]). According to the actual parameters of the boiler drum water level, the transfer function of the water flow about water level can be denoted as $G(s) = 0.0037/(30s^2 + s)$. Based on a SISO network, traditional PID controller can be denoted as $u[t] = K \cdot \Delta h[t] + C$. In fact, when a boiler is working, the rating of water level must be maintained between -0.08 and 0.08 m. So, to the error of water level Δh, the range should be $[-0.08\ \mathrm{m}, +0.08\ \mathrm{m}]$. This paper takes the interval $[-3, 3]$ as range of the level's error Δh and the direct voltage u, then the coefficient $K = 3/0.08 = 37.5$.

Then we can design the controller of polygonal FNN, and the number of fuzzy inference rules $L = 7$. Antecedent polygonal number $\tilde{X}(l)$ can be denoted as water level's error, and consequent polygonal number $\tilde{O}(l)$ can be denoted as direct voltage. We can use the single–input–single–output relationship as the above polygonal numbers, then we can get fuzzy pattern pairs of polygonal FNN $(\tilde{X}(l), \tilde{O}(l))$, $l = 1, 2, \ldots, 7$. To simplify, let $n = 4$, and randomly select actual input $\tilde{X}(l)$ and expected output $\tilde{O}(l)$ corresponding to 4-polygonal fuzzy pattern pairs (see Tables 1, 2).

From the above Tables 1, 2 showed two 4-polygonal fuzzy pattern pairs, we can draw the membership functions' curves of the actual input $\tilde{X}(l)$ and expected output $\tilde{O}(l)$ $(l = 1, 2, \ldots, 7)$ with Mathematica software in Figs. 5, 6.

We can apply the original BP algorithm, the traditional GA algorithm and the GA-BP hybrid algorithm to

Table 1 Trained 4-polygonal fuzzy pattern pairs

l	Actual input $\tilde{X}(l)$
$l = 1$	(−3.00, −3.00, −3.00, −3.00, −3.00, −2.85, −2.70, −2.60, −2.50, −2.35)
$l = 2$	(−2.65, −2.50, −2.40, −2.30, −2.15, −1.85, −1.70, −1.60, −1.50, −1.35)
$l = 3$	(−1.65, −1.50, −1.40, −1.30, −1.15, −0.85, −0.70, −0.60, −0.50, −0.35)
$l = 4$	(−0.65, −0.50, −0.40, −0.30, −0.15, 0.15, 0.30, 0.40, 0.50, 0.65)
$l = 5$	(0.35, 0.50, 0.60, 0.70, 0.85, 1.15, 1.30, 1.40, 1.50, 1.65)
$l = 6$	(1.65, 1.50, 1.60, 1.70, 1.85, 2.15, 2.30, 2.40, 2.50, 2.65)
$l = 7$	(2.65, 2.50, 2.60, 2.70, 2.85, 3.00, 3.00, 3.00, 3.00, 3.00)

Table 2 Trained 4-polygonal fuzzy pattern pairs

l	Expected output $\tilde{O}(l)$
$l = 1$	(−3.00, −3.00, −3.00, −3.00, −3.00, −2.90, −2.85, −2.70, −2.65, −2.40)
$l = 2$	(−2.60, −2.35, −2.30, −2.15, −2.10, −1.90, −1.85, −1.70, −1.65, −1.40)
$l = 3$	(−1.60, −1.35, −1.30, −1.15, −1.10, −0.90, −0.85, −0.70, −0.65, −0.40)
$l = 4$	(−0.60, −0.35, −0.30, −0.15, −0.10, 0.10, 0.15, 0.30, 0.35, 0.60)
$l = 5$	(0.40, 0.65, 0.70, 0.85, 0.90, 1.10, 1.15, 1.30, 1.35, 1.60)
$l = 6$	(1.40, 1.65, 1.70, 1.85, 1.90, 2.10, 2.15, 2.30, 2.35, 2.60)
$l = 7$	(2.40, 2.65, 2.70, 2.85, 2.90, 3.00, 3.00, 3.00, 3.00, 3.00)

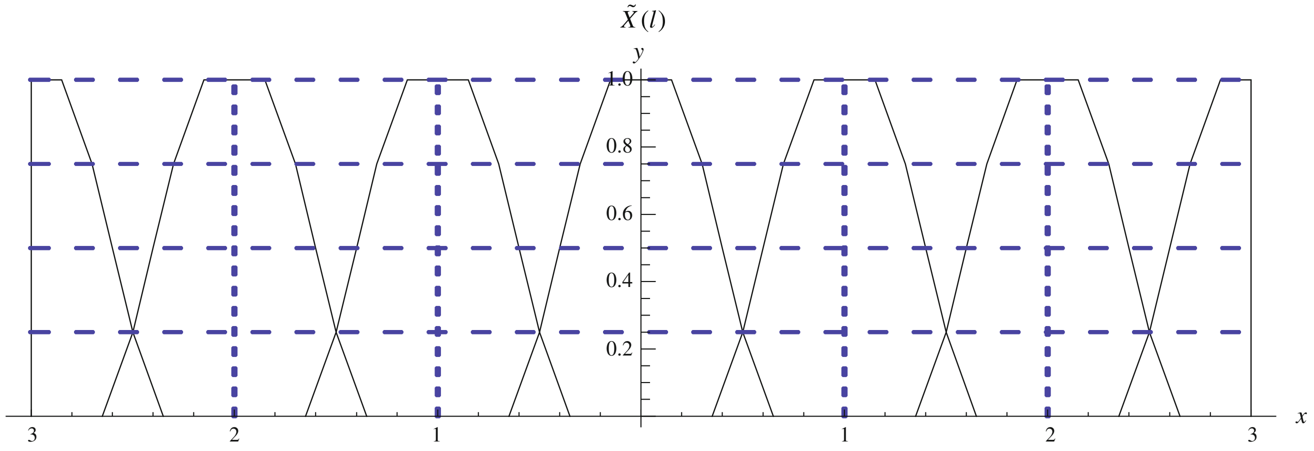


Fig. 5 Antecedent 4-polygonal fuzzy numbers

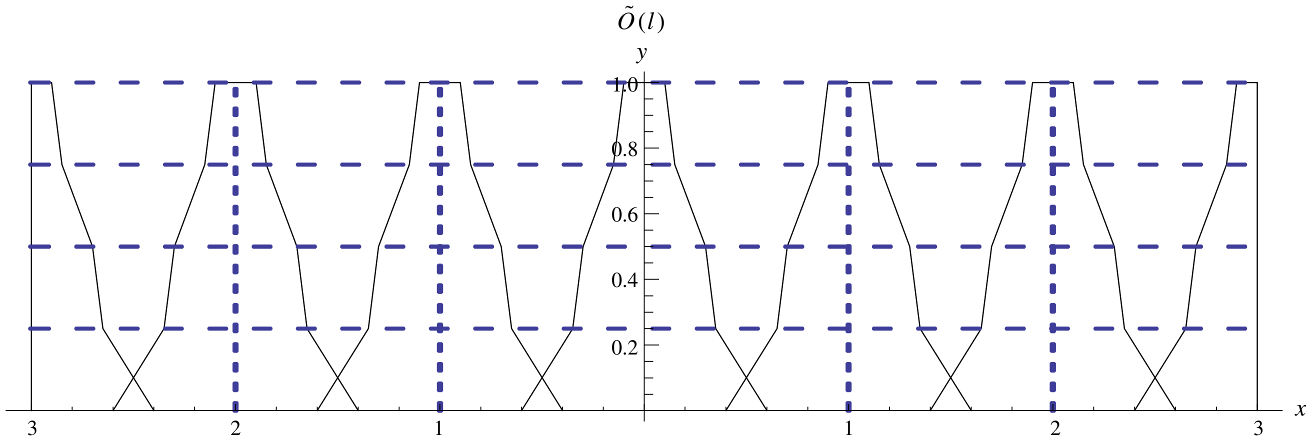


Fig. 6 Consequent 2-polygonal fuzzy numbers

parameters optimization based on polygonal FNN. Let transfer function $\sigma(x) = 0, x < 0$; $\sigma(x) = x/(1 + x^2),\ x \geq 0$, and error accuracy $\varepsilon = 10^{-3}$, searching accuracy $\gamma = 10^{-6}$, total of groups $m_0 = 40$, where the number of hidden layers is 14. If error function $E(w)$ satisfies the condition or the algorithm's iteration is stopped, we can get the following optimization results (see Table 3).

From Table 3, we can see that the GA-BP hybrid algorithm can reduce the number of iterations to 174, and greatly reduce the number of local minimum. This algorithm doesn't only overcome slow convergence of the original BP algorithm which is easy to fall into local minimum, but also enhances the ability of the GA algorithm's global searching to improve the rate of polygonal FNN's parameters optimization. The optimized parameters of the above three algorithms are applied to polygonal FNN's controller in the boiler drum water level control system, and we can get the system's simulation results by comparing PID controller (see Fig. 7).

Clearly, from Fig. 7, we can get that the system of the simulation curve's amplitude is small, and the system responses fast, control accuracy is high, and it has the best optimization result by parameters optimization of the GA-BP hybrid algorithm based on the polygonal FNN's controller. In addition, comparing with the original BP and the traditional GA algorithm, parameters optimization is improved, and reduces the system's overshoot and responding time greatly by the GA-BP hybrid algorithm. Then the system can be adjusted to a steady state in short period of time, and also improved its stability. The result shows that parameters optimization of the GA-BP hybrid algorithm for polygonal FNN is better than others. And the optimization doesn't only get rid of the dependence of the original BP algorithm's initial points and local convergence, but also overcomes the random and probabilistic problem of the traditional GA algorithm. Therefore, the optimization can lay the foundation to study the good performance of polygonal FNN in the future.

Table 3 Comparison of parameters optimization results among three algorithms

Algorithm	Iterations	Convergence	Number of local minimum
BP	532	392 s	9
GA	403	565 s	2
GA-BP	174	430 s	1

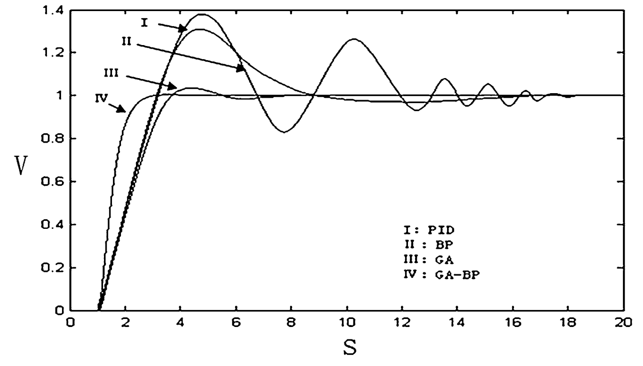


Fig. 7 The simulation curve of boiler drum water level control system

6 Conclusion

In this article, based on the expansive operations of polygonal fuzzy numbers, we combine the BP algorithm with the GA algorithm, and design a GA-BP hybrid learning algorithm for polygonal FNN. Finally, we prove the effectiveness of the GA-BP hybrid algorithm by simulation examples. In fact, because the algorithm is complex, it results in its convergence's time to be slightly longer than the BP algorithm. In addition, we don't add the other convergence criteria to solve the global optimal value, and the GA algorithm has blindness and probabilistic problem. Hence, the algorithm maybe doesn't reach the optimal solution, but just approximate the optimal value. So how to design simple and practical algorithm for polygonal FNN is a further research. For example, join A-G criteria of the dynamic convergence condition to solve the global optimal value. In addition, it is worth studying further to improve the parameters' iteration formula of the BP algorithm appropriately.

References

1. Buckley JJ, Hayashi Y (1994) Fuzzy neural networks: a survey. Fuzzy Sets Syst 66(1):1–13
2. Buckley JJ, Hayashi Y (1994) Can fuzzy neural nets approximate continuous fuzzy function. Fuzzy Sets Syst 61(1):43–51
3. Liu PY (2002) A new fuzzy neural network and its approximation capability. Sci China (E) 32(1):76–86
4. Liu PY (2002) Fuzzy neural network theory and applications [D]. Beijing Normal University, Beijing
5. He CM, Ye YP (2011) Evolution computation based learning algorithms of polygonal fuzzy neural networks. Int J Intell Syst 26(4):340–352
6. Wang GJ, Li XP (2011) Universal approximation of polygonal fuzzy neural networks in sense of K-integral norms. Sci China Inf Sci 54(11):2307–2323
7. He Y, Wang GJ (2012) The conjugate gradient algorithm of the polygonal fuzzy neural networks. Acta Electronica Sinica 40(10):2079–2084
8. Sui XL, Wang GJ (2012) Influence of perturbations of training pattern pairs on stability of polygonal fuzzy neural network. Pattern Recog Artif Intell 26(6):928–936
9. Chen Chuen-Jyh (2012) Structural vibration suppression by using neural classifier with genetic algorithm. Int J Mach Learn Cybernet 3(3):215–221
10. Zhang SW, Wang LL, Chen YP (2008) A fuzzy neural network controller based on GA-BP hybrid algorithm. Control Theor Appl 30(2):3–5
11. Dong LT, Robert M (2010) Genetic Algorithm-Neural Network (GANN): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection. Int J Mach Learn Cybernet 1(4):75–87
12. Tobias F, Trent K, Frank N (2013) Weighted preferences in evolutionary multi-objective optimization. Int J Mach Learn Cybernet 4(2):139–148
13. Zhang JL, Zhang HG, Luo YH (2013) Nearly optimal control scheme using adaptive dynamic programming based on generalized fuzzy hyperbolic model. Acta Automatica Sinica 39(2):142–149
14. Mahapatra GS, Mandal TK, Samanta GP (2011) A production inventory model with fuzzy coefficients using parametric geometric programming approach. Int J Mach Learn Cybernet 2(2):99–105
15. Li QL, Lei HM, Xu XL (2010) Training self-organizing fuzzy neural networks with unscented kalman filter. Syst Eng Electron 32(5):1029–1033
16. Xue H, Li X, Ma HX (2009) Fuzzy dependent-chance programming using ant colony optimization algorithm and its convergence. Acta Automatica Sinica 35(7):959–964
17. Cao YJ, Wu QH (1999) Teaching genetic algorithm using MATLAB. Int J Electr Eng Educ 36(2):139–153
18. Qiao JF, Li M, Liu J (2010) A fast pruning algorithm for neural network. Acta Electronica Sinica 38(4):830–834
19. Kao CH, Hsu CF, Don HS (2012) Design of an adaptive self-organizing fuzzy neural network controller for uncertain nonlinear chaotic systems. Neural Comput Appl 12:1243–1253
20. Sui D, Fang X (2011) Optimization of PID controller parameters based on neural network. Comput Simul 28(8):177–180
21. Lin CM, Li MC, Ting AB, Lin MH (2011) A robust self-learning PID control system design for nonlinear systems using a particle swarm optimization algorithm. Int J Mach Learn Cybernet 2(4):225–234