Scientific Research and Essays Vol. 6(15), pp. 3109-3121, 11 August, 2011
Available online at http://www.academicjournals.org/SRE
DOI: 10.5897/SRE10.691
ISSN 1992-2248 ©2011 Academic Journals

Full Length Research Paper

Application of particle swarm optimization algorithm-based fuzzy BP neural network for target damage assessment

Hui Yuan[1], Jun Zhi[1,2]* and Jianyong Liu[1]

[1]Engineering Institute of Corps of Engineers, PLA University of Science and Technology, Nanjing 210007, China.
[2]Zhenjiang Watercraft College of PLA, Zhenjiang 212003, China.

Accepted 21 April, 2011

It has proposed a kind of hybrid method based on intuitionistic fuzzy set theory and particle swarm optimization (PSO) algorithm-based neural network (NN). We apply it to the integrative damage effect assessment of battlefield target. Firstly, we improve PSO algorithm, propose the adaptive inertia factor and excellence selection mechanism, introduce inter-partition particle swarm initialization and simulated annealing mutation mechanism. Secondly, we use the improved PSO algorithm to optimize the initial weights and thresholds of neural network to improve the network structure. Thirdly, we carry on the fusion of multiple different neural networks based on intuitionistic fuzzy set theory. The purpose of this study is to determine the weight of different neural networks, and synthesize their assessment result has the final output according to the weight. The effectiveness and reasonableness of algorithm are improved by simulation results.

Key words: PSO algorithm, fuzzy neural network, target damage assessment.

INTRODUCTION

BP NN is a sort of multilayer feed-forward single direct propagation network model, divided into input layer, hidden layer and output layer. In order to get a better assessment effect of NN, we must construct the network well. Whether the value of initial weight and threshold is reasonable or not will influence the final iteration effect of NN algorithm. When the nodes of network are relatively more, selection of initial weight and threshold will meet similar problems as “combination explosion”, a NP hard problem. It is very difficult to solve the problem through traditional experiment method. So we adopt PSO algorithm to optimize the initial weigh and threshold of neural network.While NN is regarded as an evaluation method, correspondent to different samples, comparison between neural networks with different structures, different training and different optimization always show differences. While inputting training samples to complete the network learning and training in the past, this kind of difference between samples is always not considered (Fan, 2004). If we adopt two kinds of different neural networks and divide training samples into two parts to train the networks, we can often find that they correspond to different samples of two parts, the comparison difference is great. We try to consider dividing training samples into groups according to certain method while using training samples to train different neural networks (Wang and Gao, 2001; Hong, 2000). In this way, while regarding assessment problem as decision problem, different neural networks can be regarded as different “decision schemes”, grouped samples can the regarded as “attribute” of decision problem (Mikhailov et al., 2005). Specifically, we can consider carrying on the fusion of multiple neural networks with intuitionistic fuzzy set theory to improve the accuracy of assessment.

ASSESSMENT MODEL BASED ON BP NN

Set of input layer

Set number of input nodes of neural network network as

*Corresponding author. E-mail: njzhijun@163.com.

m_1, the sample vector of input layer is $P=(p_1,p_2,\cdots,p_{m_1})$. $p_i(i=1,2,\cdots m)$ denote the i th index value of sample. The linking weight between i th neuron of input layer and j th neuron of hidden layer is w_{1ij}. Its adjustment algorithm is:

$$w_{1ij}(k+1)=w_{1ij}(k)-\eta\frac{\partial E}{\partial p_i}f_{1j}(w_{1ij}\sum_{i=1}^{m_1}p_i+b_{1j}) \quad 1$$

It is a kind of steepest descent algorithm. It denotes the algorithm change of linking weight w_{1ij} from k th generation to k+1 th generation. E is error function. f_{1j} is the output function of j th neuron of hidden layer, namely the active function of input layer. b_{1j} is the threshold of j th neuron of hidden layer. η is the learning step, usually it's a constant. But in different network training periods, the same learning step may bring different training effects, difference is always great. So we consider introducing method of self-adaptive study step:

$$\eta(k+1)=\begin{cases}1.15\eta(k) & if\ E(k+1)<E(k)\\ 0.7\eta(k) & if\ E(k+1)>1.14E(k)\\ \eta(k) & other\end{cases} \quad 2$$

Equation 2 is to check whether weight modification reduces error function in every iteration step of algorithm. If it does, the learning step is small, we should increase some, otherwise reduce some. Similarly, we can draw the adjustment algorithm of threshold value as follows:

$$b_{1j}(k+1)=b_{1j}(k)-\eta\frac{\partial E}{\partial p_i} \quad 3$$

Set of hidden layer

Number of hidden neurons is having greater influence on learning ability and generalization capability. If the number of neurons is more, convergence speed of algorithm will be reduced and the computation is too complicated. If the number of neurons is less, the algorithm can not reach certain precision; it is more difficult to find efficient solutions. We can get the right number of hidden neurons by repeated experiments according to specific problems, that is m_2. Output function f_{1j} of the j th neuron of hidden layer is:

$$f(x)=\frac{1-e^{-2x}}{1+e^{-2x}} \quad 4$$

The linking weight between j th hidden neuron and h th output neuron is w_{2jh}, and the threshold of h th output neuron is b_{2h}. Their adjustment method adopts above-mentioned steepest descent algorithm and learning step is also self-adaptive:

$$w_{2jh}(k+1)=w_{2jh}(k)-\eta\frac{\partial E}{\partial f_{1j}}f_{2h}(w_{2jh}\sum_{j=1}^{m_2}f_{1j}+b_{2h}) \quad 5$$

$$b_{2h}(k+1)=b_{2h}(k)-\eta\frac{\partial E}{\partial f_{1j}} \quad 6$$

In addition, we can consider that introduce the so-called "momentum" on the linking weight adjustment between two layers:

$$w(k+1)=w(k)+\Delta w+\beta\Delta w' \quad 7$$

Where, Δw is basic adjustment, $\beta\Delta w'$ is namely the momentum, and β is momentum coefficient within the range of [0.01, 0.9]. We can find suitable β value to make training effect of network better by repeated experiments.

Set of output layer

Number of output neurons is m_3, that is the solution number. The output function of h th output neuron is f_{2h}. It adopts sigmoid function:

$$f(x)=\frac{1}{1+e^{-x}} \quad 8$$

We can define the error function of network through output function on the output layer:

$$E=\frac{1}{2}\sum_{h=1}^{m_3}(t_h-f_{2h})^2 \quad 9$$

Where, t_h is anticipant output of the h th neuron. Error value can be regarded as the convergence condition of NN algorithm. Namely when error value is less than certain positive constant, we will end the iteration.

APPLYING PSO ALGORITHM TO THE OPTIMIZATION OF NN

PSO algorithm

Common used PSO algorithm is called PSO algorithm of

inertia weight. Its basic iteration formula is:

$$v_{id} = w \times v_{id} + c_1 \times r_1 \times (P_{id} - x_{id}) + c_2 \times r_2 \times (P_{gd} - x_{id}) \quad 10$$

$$x_{id} = x_{id} + v_{id} \quad 11$$

Where, w is inertia weight, c_1 and c_2 are positive constants, r_1 and r_2 are random numbers within the range of [0, 1]. Equation (10) denotes the change of particle speed. Equation (11) denotes the change of particle position. In the PSO algorithm, every particle in the swarm stands for a selected subset of features, it can be expressed as $X_i = (x_{i1}, x_{i2}, \cdots, x_{id}, \cdots, x_{iD})$.

Improvement of PSO algorithm

Inter-partition particle swarm initialization

If distribution of particle swarm in the solution domain is sufficient and even, process in solving can get better effect. We introduce the division method of solution domain to carry on the initialization of particle swarm. Namely confirm particle positions according to the division of intervals. Set the position of i th particle as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$. If particle number is *m*, we can divide the solution domain into *m* different regions. Each region produces a particle, so generation area of the i th particle is as follows:

$$X_{i,k} \in [a_k + f_k(b_k - a_k), a_k + f_{k+1}(b_k - a_k)] \quad k = 1, 2, \cdots, D$$

$$\begin{cases} f_k = (i-1) \bmod C^{D-k} \\ f_D = (i-1) - \sum_{j=1}^{D-1} f_j C^{D-j} \\ C = \sqrt[D]{m} \end{cases} \quad 12$$

We can generate initial position of the i th particle according to Equation (13).

$$X_{i,k} = a_k + f_k(b_k - a_k) + \frac{1}{\sqrt[D]{m}}(b_k - a_k) r \quad 13$$

Where, *r* is a random number within [0, 1].

Adaptive inertia factors

At the later iteration stage of algorithm, most particles have already trended to approach the optimum position. If the inertia factor is still very great at this moment, it is very apt to deviate from the optimum position and very difficult to orient effectively. So we consider making the inertia of particle smaller when it is close to the optimum position. In this way the localization of particle is higher in precision, it can approach the optimum position more effectively (Chen et al., 2010; Liang and Yang, 2009).The concrete realization method is as follows:

$$w = \frac{\sum_{j=1}^{D} (X_{gj} - X_j)}{\sum_{j=1}^{D} X_{gj}} \times w_0 \quad 14$$

Where w_0 is inertia factor, we usually fetch about 0.8, X_{gj} is the j th dimension value of present optimum position vector. X_j is the j th dimension value of present position vector.

Selection mechanism

At the early iteration period of algorithm, the differences between particle positions are great. It is better to distinguish elite individual according to fitness. At the late iteration period of algorithm, the differences between particle positions are minor; it is difficult to distinguish the elite individual and the selection operation is meaningless (Song and Chen, 2007; Jing and Ren, 2006). So at different periods of algorithm, while calculating fitness of particles, we adopt different types of fitness functions to improve the effects of selection. The concrete method is as follows:

Fitness function

$$J = \begin{cases} \frac{1}{Z} & when \quad \varepsilon > kt \\ \frac{1}{e^{\alpha Z}} & when \quad \varepsilon < kt \\ \frac{1}{e^{\beta Z}} & when \quad \varepsilon < t \end{cases} \quad 15$$

Where, Z is the objective function value with calculation; ε is the convergent degree of algorithm, t is obtained with relatively small numbers in general, α, β and k are positive constants, $\alpha < \beta$, they can be adjusted through tests. We can divide the iteration process of PSO algorithm into three different periods according to the convergent degree of algorithm, they are $\varepsilon > kt$, $\varepsilon < kt$ and $\varepsilon < t$. We can adopt different types of function according to Equation (15), and distinguish different particle individuals effectively to some extent.

Simulated annealing mechanism

We adopt the principle of simulated annealing to carry on the neighbor-region search of each particle at every iteration process of algorithm. Set original particle individual as *X*. Generate a new individual X' to replace the original individual *X* with probability *p*. The probability is calculated as follows:

$$p=\begin{cases}1 & f(x')\geq f(x)\\ \exp(\dfrac{f(x)-f(x')}{T}) & f(x')<f(x)\end{cases} \quad (16)$$

Where, *T* is current temperature, at each iterative operation of PSO algorithm, set a larger initial temperature $T(t_0)$, and then carry on the method of $T(t+1)=\gamma\times T(t)$ to reduce temperature. γ is a positive number slightly less than 1. We can find that substitution probability decreases as temperature decreases. The concrete method of generating new individual X' is as follows:

$$\begin{cases}x'_{id}=x_{id}+\alpha\Delta x_d\\ \Delta x_d \sim N(0,\delta_d)\end{cases} \quad (17)$$

Where, $N(0,\delta_d)$ denotes that mean value is 0, variance is the normal distribution of ${\delta_d}^2$, and Δx component are mutual exclusive.

Initial weight and threshold optimization

Mapping weight and threshold value of network as a group of particles, namely one particle is a solution for a group of initial value. Particle number is namely the dimension of solution domain:

$$D=m_1*m_2+m_2*m_3+m_1+m_2 \quad (18)$$

m_1*m_2 is the linking weight number between input layer and hidden layer, m_2*m_3 is the linking weight number between output layer and hidden layer, m_1 is the threshold number of hidden layer, m_2 is the threshold number of output layer (Mu and Shen, 2009; Chang et al., 2009). Real coding is used. Fitness function of each particle adopts the reciprocal of error function. The larger the value of fitness function is, the finer the performance of particle is. Through structure improvement of NN, we apply PSO algorithm to optimize the initial weight and threshold of NN. Concrete procedures are as follows:

Step 1: Sample pretreatment. Collect data; analyze the factors affecting assessment effect and found sample data database. The samples are handled to be normalization.
Step 2: Determine the input, output, number of neuron and error function, etc., according to characteristics of samples.
Step 3: Set initial values of weight and threshold of neural network, and iterations, etc.
Step 4: Initialize the initial position, speed, inertia weight, etc., of particles and determine the population scale, restraint condition, etc. Calculate fitness of particles according to training samples and begin to iterate. Update the velocity and position of particles. Output the result finally, and determine initial weight and threshold of neural network.
Step 5: Input training samples again, and perform the iterative optimization of neural network, finally determine network structure.
Step 6: Input the sample to be assessed and output the result.

FUSION ALGORITHM BASED ON INTUITIONISTIC FUZZY SETS

Intuitionistic fuzzy sets is firstly structuring intuitionistic judgment matrix and intuitionistic fuzzy decision-making matrix according to characteristic of decision-making problem (Xu, 2004, 2007), secondly calculating weight of schemes or indexes according to certain method, thirdly ranking the decision-making schemes and selecting the most desirable ones.

Intuitionistic fuzzy sets

Definition 1. Set *X* as nonempty class, *F*= $\{\langle x,\mu_F(x),v_F(x)\rangle \mid x\in X\}$ is called intuitionistic fuzzy sets. $\mu_F(x)$ and $v_F(x)$ are respectively membership and non-membership of *x* belongs to *X*. $\mu_F \rightarrow$[0, 1], $v_F \rightarrow$[0, 1], $0\leq \mu_F(x)+v_F(x)\leq 1, \forall\ x\in X$.
Usually we take the general form of intuitionistic fuzzy numbers abridged for $\alpha=(\mu_\alpha,v_\alpha)$, $0\leq \mu_\alpha+v_\alpha\leq 1$. Its value can be calculated through score function $s(\alpha)=\mu_\alpha-v_\alpha, s(\alpha)\in$ [-1,1]. The score value $s(\alpha)$ can be an important index to measure the size of intuitionistic fuzzy numbers α.

Intuitionistic judgment matrix

Definition 2. Set $Y=\{y_1,y_2,\cdots,y_n\}$ as the scheme collection, decision-maker carry on comparison between

n schemes and construct a judgment matrix $B = (b_{ij})_{n\times n}$, where $b_{ij} = (\mu_{ij}, v_{ij})(i, j = 1,2,\cdots,n)$. μ_{ij} denotes the preferences of x_i when decision-maker compare x_i and x_j. v_{ij} denotes the preferences of x_j. If $\mu_{ij} \in [0, 1], v_{ij} \in [0,1], \mu_{ji} = v_{ij}$, $v_{ji} = \mu_{ij}$, $\mu_{ii} = v_{ii} = 0.5$, $\mu_{ij} + v_{ij} \le 1(i, j = 1,2,\cdots,n)$, *B* is called intuitionistic judgment matrix.

When multiple neural networks to be carried on the evaluation, $Y = \{y_1, y_2, \cdots, y_n\}$ is a collection for neural network, through the paired comparison between *n* neural networks, we can construct a judgment matrix $B = (b_{ij})_{n\times n}$, $b_{ij} = (\mu_{ij}, v_{ij})(i, j = 1,2,\cdots,n)$. μ_{ij} denotes the relative credibility of NN x_i compared to x_j. v_{ij} denotes the relative credibility of NN x_j compared with x_i. Calculation formula of the relative credibility is as follows:

$$\mu_{ij} = \frac{E_j}{E_i},\ v_{ij} = \frac{E_i}{E_j},\ i, j = 1,2,\cdots n \qquad 19$$

Where, E_i and E_j respectively denotes error value of the neural network, *i* and *j* after continuing learning and training the overall sample. Thus we have constructed an intuitionistic judgment matrix of multiple neural networks based on overall samples. According to definition 2, we can transform the element $b_{ij} = (\mu_{ij}, v_{ij})$ of intuitionistic judgment matrix into an interval form $\dot{b}_{ij} = [\mu_{ij}, 1 - v_{ij}]$. It makes intuitionistic judgment matrix $B = (b_{ij})_{n\times n}$ equal to an interval complementary judgment matrix $\dot{B} = (\dot{b}_{ij})_{n\times n}$, where $\dot{b}_{ij} = [\dot{b}_{ij}^-, \dot{b}_{ij}^+] = [\mu_{ij}, 1 - v_{ij}]$ and $\dot{b}_{ij}^- + \dot{b}_{ji}^+ = \dot{b}_{ij}^+ + \dot{b}_{ji}^- = 1, \dot{b}_{ij}^+ \ge \dot{b}_{ij}^- \ge 0, \dot{b}_{ii}^+ = \dot{b}_{ii}^- = 0.5, i, j = 1,2\cdots,n.$ So the element of intuitionistic judgment matrix can be expressed by $b_{ij} = [\mu_{ij}, 1 - v_{ij}]$. In particular, if $\mu_{ij} + v_{ij} = 1$, the intuitionistic judgment matrix $B = (b_{ij})_{n\times n}$ becomes complementary judgment matrix.

Intuitionistic fuzzy decision-making matrix

We can construct an intuitionistic fuzzy decision-making matrix similar to the intuitionistic judgment matrix according to the satisfactory degree of the decision scheme to the decision problem attributes.

$A = \{A_1, A_2, \cdots, A_n\}$ is a collection for neural network, $G = \{G_1, G_2, \cdots, G_m\}$ is a collection of grouped samples, and $w = (w_1, w_2, \cdots, w_m)^T \in H$ is the weight vector of the grouped samples. In the decision-making matrix $R = (r_{ij})_{n\times m}$, $r_{ij} = (\alpha_{ij}, \beta_{ij})$, α_{ij} denotes the credibility of neural network A_i to grouped samples G_j, β_{ij} denotes the none credibility of neural network A_i to grouped samples G_j. $0 \le \alpha_{ij} + \beta_{ij} \le 1$, $i = 1,2,\cdots,m; j = 1,2,\cdots,n$. Calculation formulas of credibility and incredibility are as follows:

$$\theta_i = \frac{C - E_i}{\sum_{k=1}^{n}(C - E_k)}, \phi_i = \frac{E_i}{\sum_{k=1}^{n} E_k},\ i = 1,2,\cdots n \qquad 20$$

Where, E_i denotes error value of the neural network *i* after going on learning and training to the grouped samples. *C* is a positive constant, we usually fetch one. Thus we have constructed an intuitionistic fuzzy decision-making matrix of multiple neural networks based on grouped samples. We can make use of score function to calculate and get the score matrix of fuzzy decision-making matrix, that is $S = (s(r_{ij}))_{n\times m}$, $s(r_{ij}) = s(\alpha_{ij}, \beta_{ij}) = \alpha_{ij} - \beta_{ij}$, $s(r_{ij}) \in [-1,1], i = 1,2,\cdots,n; j = 1,2,\cdots,m$.

Then use formula (6) to get $\bar{S} = (\bar{s}(r_{ij}))_{n\times m}$ utilizing score matrix $S = (s(r_{ij}))_{n\times m}$:

$$\bar{s}(r_{ij}) = \frac{s(r_{ij}) - \min_i\{s(r_{ij})\}}{\max_i\{s(r_{ij})\} - \min_i\{s(r_{ij})\}} \qquad 21$$

In line with the standardized score matrix $\bar{S} = (\bar{s}(r_{ij}))_{n\times m}$, the synthetic score of each neural network can be shown as follows:

$$\bar{s}(r_i) = \sum_{j=1}^{m} w_j \bar{s}(r_{ij}),\ i = 1,2,\cdots,n \qquad 22$$

Where, the weight information is completely unknown, so we need to set up certain model to get the solution of attribute weight.

Additive consistent linear programming model

In order to try to get the attribute weight, the additive consistent linear programming model is introduced. Combining with the theory of intuitionistic fuzzy sets, we provide the following definition of additive consistent linear intuitionistic judgment matrix.

Definition 3. Set $C=(c_{ij})_{n\times n}$ as complementary judgment matrix. If $c_{ij}=c_{ik}-c_{jk}+0.5, i,j,k=1,2,\cdots,n, C$ is called the additive consistent complementary judgment matrix. Set $w=(w_1,w_2,\cdots,w_m)^T$ as the ranking vector of additive consistent complementary judgment matrix *C*. Element c_{ij} in *C* can be written as $c_{ij}=0.5(w_i-w_j+1)$.

Definition 4. Set $B=(b_{ij})_{n\times n}$ as intuitionistic judgment matrix, $b_{ij}=[\mu_{ij},1-v_{ij}]$ $(i,j=1,2,\cdots,n)$. If there exists vector $w=(w_1,w_2,\cdots,w_n)^T$ making $\mu_{ij}\le 0.5(w_i-w_j+1)\le 1-v_{ij}$, $i,j=1,2,\cdots,n$ $w_j\ge 0$, $\sum_{j=1}^{n}w_j=1$. We call *B* additive consistent judgment matrix. In order to make the decision information identical, we use the synthetic score of the entire neural network $A_i\ (i=1,2,\cdots n)$ to construct additive consistent linear complementary judgment matrix $\overline{B}=(\overline{b}_{ij})_{n\times n}$, where $\overline{b}_{ij}=0.5(\overline{s}(r_i)-\overline{s}(r_j)+1)$, $i,j=1,2,\cdots,n$. In order to obtain the attribute weight vector $w=(w_1,w_2,\cdots,w_n)^T$, the following linear programming model.

$$w_k^- = \min w_k \quad 23$$

$$s.t.\begin{cases} 0.5(\sum_{k=1}^{m}w_k(\overline{s}(r_{ik})-\overline{s}(r_{jk}))+1)+d_{ij}^-\ge\mu_{ij} \\ 0.5(\sum_{k=1}^{m}w_k(\overline{s}(r_{ik})-\overline{s}(r_{jk}))+1)-d_{ij}^+\le 1-v_{ij} \\ w=(w_1,w_2,\cdots,w_n)^T\in H, w_i\ge 0, \sum_{i=1}^{n}w_i=1 \end{cases}$$

$$w_k^+ = \max w_k \quad 24$$

$$s.t.\begin{cases} 0.5(\sum_{k=1}^{m}w_k(\overline{s}(r_{ik})-\overline{s}(r_{jk}))+1)+d_{ij}^-\ge\mu_{ij} \\ 0.5(\sum_{k=1}^{m}w_k(\overline{s}(r_{ik})-\overline{s}(r_{jk}))+1)-d_{ij}^+\le 1-v_{ij} \\ w=(w_1,w_2,\cdots,w_n)^T\in H, w_i\ge 0, \sum_{i=1}^{n}w_i=1 \end{cases}$$

Where, d_{ij}^- and d_{ij}^+ are not nonnegative real numbers, they are introduced deviation variable while the additive consistent linear complementary judgment matrix $\overline{B}$ disaccords with the intuitionistic judgment matrix constructed by the comparison between *n* neural networks. When matrix $\overline{B}$ and B go all the way, d_{ij}^- and d_{ij}^+ are expressed as zero. We can judge whether the matrix $\overline{B}$ and B go all the way with Equation (25):

$$\mu_{ij}\le 0.5(\sum_{k=1}^{m}w_k(\overline{s}(r_{ik})-\overline{s}(r_{jk}))+1)\le 1-v_{ij}, \quad 25$$

The weight vector quantity that satisfies the condition is more than one. So every weight w_k belongs to certain zone. Through solving the model (23) and (24), we can acquire the collection of attribute weight vector:

$$\Delta_1=\{w=(w_1,w_2,\cdots,w_m)^T \mid w_k\in[w_k^-,w_k^+],$$
$$w_k\ge 0, k=1,2,\cdots,m, \sum_{k=1}^{m}w_k=1\}$$

FUSION ALGORITHM DESIGN

Step 1: Set up the following linear programming model to get the weight vector quantity of optimum attribute on the basis of the obtained interval vector quantity of attribute weight (Zhou and Duan et al., 2006):

$$\varphi = Max\sum_{i=1}^{n}\sum_{j=1}^{m}(1-\beta_{ij}-\alpha_{ij})w_j \quad 26$$
$$s.t.\quad w=(w_1,w_2,\cdots,w_n)^T\in\Delta$$

The obtained weight vector quantity of optimum attribute is as follows:

$$w^*=(w_1^*,w_2^*,\cdots,w_n^*)^T$$

Step 2: Calculate the synthetic score of each neural network.

$$z_i(w^*)=[z_i^-(w^*),z_i^+(w^*)](i=1,2,\cdots,m)$$

Where $z_i^-(w^*)=\sum_{j=1}^{m}w_j^*\alpha_{ij}$, $z_i^+(w^*)=\sum_{j=1}^{m}w_j^*(1-\beta_{ij})$

Step 3: Construct the credibility matrix $P=(p_{ij})_{n\times n}$

Table 1. Grade standard of damage degree.

Damage degree	No damage	Slight damage	Low-grade damage	Medium-sized damage	Little serious damage	Serious damage	Scrap
Grade standard	0~0.05	0.05~0.2	0.2~0.4	0.4~0.6	0.6~0.75	0.75~0.95	0.95~1

based on the comparison between the comprehensive score $z_i(w^*)(i=1,2,\cdots,n)$ of every neural network.

$$p_{ij}=p(z_i(w^*)\geq z_j(w^*))$$
$$=\max\left\{1-\max\left(\frac{z_j^+(w^*)-z_i^-(w^*)}{z_i^+(w^*)-z_i^-(w^*)+z_j^+(w^*)-z_j^-(w^*)},0\right),0\right\}$$

Step 4: Obtain the weight vector quantity $\omega=(\omega_1,\omega_2,\cdots,\omega_n)$ of neural network according to the following formula:

$$\omega_i=\frac{1}{n(n-1)}\left(\sum_{j=1}^{n}p_{ij}+\frac{n}{2}-1\right) \quad 27$$

Step 5: Finally fuse the assessed output vector $O=(o_1,o_2,\cdots,o_n)$ of *n* neural networks:

$$R=\sum_{i=1}^{n}o_i*\omega_i \quad 28$$

To be precise, *R* is the weighted synthetic outputs of multiple neural networks.

INSTANCE ANALYSES

Selection and organizing of samples

When we have determined the input and output samples, we should carry on normalization to the samples (Chau et al., 2007). The method is generally to limit the data between [0, 1] or [-1, 1]. Adopt the following formula to change data into the range of [0, 1]:

$$\bar{x}_i=\frac{x_i-x_{\min}}{x_{\max}-x_{\min}} \quad 29$$

Adopt the following formula to change data into the range of [-1, 1]:

$$x_{\min}=\frac{x_{\max}-x_{\min}}{2} \quad 30$$

$$\bar{x}_i=\frac{x_i-x_{mid}}{x_{\max}-x_{\min}} \quad 31$$

Where, x_i denotes the input or output data, $x_{\min}$ is the minimum, $x_{\max}$ is the maximum, x_{mid} is the median of data change range.

Group of training samples

We use 122 mm shrapnel and 152 mm shrapnel as the types of artillery weapon, and regard 122 howitzer, 152 cannon howitzer and 130 cannon as the battlefield target. We set up the damage assessment model of the artillery to the battlefield target, and use the fuzzy neural network to carry on the damage effect assessment. Table 1 is the damage assessment standard created by experts, corresponding different destruction degree of targets to set up different damage value range. There are thirty six training samples sifted for the neural network train in Tables 2 to 5. Every nine training samples will be divided into one group and we get a, b, c, d four groups. There are four test samples in Table 6. The concrete explanations for the tables are as follows:

For X_1, 0 denotes 122 mm shrapnel, 1 denotes 152 mm shrapnel, For X_2, the unit is m, For X_3, the value range is $[-\pi/2,\pi/2]$, For X_4, the value range is $[0,\pi/2]$, For X_5, the value range is $[0,2\pi]$, For X_6, 0 denotes 122 mm howitzer, 1 denotes 152 mm cannon howitzer, and 2 denotes 130 mm cannon, For X_7, 0 denotes that the target has no shelter, 1 denotes half shelter, 2 denote simple shelter and 3 denote firm shelter.

Error count of neural network training

Adopt three kinds of neural networks with different structures, which are A_1, A_2 and A_3. The numbers of the latent layer of neurons are respectively set as 10 , 12 , 14.We can receive four collections G1, G2, G3 and G4 from the grouped samples a, b, c and d. G1 includes grouped samples a, b, c; G2 includes grouped samples a, b, d; G3 includes grouped samples a, c, d; G4 includes grouped samples b, c, and d. In addition, collection G includes all the grouped samples a, b, c, d. Collection G1, G2, G3, G4 and

Table 2. Neural network training group samples A.

Serial	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Score	Destruction degree
1	0.14	3.6	0.69	0	1	0	0.36	0.85	Serious damage
2	-0.1	10.1	5.26	1	2	1	0.35	0.12	Slight damage
3	-0.02	22	4.8	1	2	1	0.35	0.07	Slight damage
4	0.12	8.2	0.9	0	0	2	0.37	0.3	Low-grade damage
5	-0.04	12	0.63	1	2	1	0.35	0.1	Slight damage
6	0.13	9	5.49	1	2	1	0.35	0.16	Slight damage
7	0.08	13.2	5.66	1	0	2	0.36	0	No damage
8	0.08	12.7	0.28	1	2	1	0.35	0.42	Medium-sized damage
9	0.13	10.8	1.11	1	2	1	0.35	0.1	Slight damage

Table 3. Neural network training group samples B.

Serial	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Score	Destruction degree
10	0.13	6.6	5.49	1	2	1	0.35	0.18	Slight damage
11	0.06	7.8	5.17	0	1	0	0.36	0.15	Slight damage
12	0.05	8.6	4.54	1	2	1	0.34	0.5	Medium-sized damage
13	-0.02	14.3	1.47	1	2	1	0.34	0.42	Medium-sized damage
14	0	0	0	1	0	2	0.36	1	Scrap
15	0.09	16.7	0.79	1	2	1	0.35	0.05	Slight damage
16	0.07	9.1	0.51	1	2	1	0.34	0.45	Medium-sized damage
17	0.06	17.5	5.04	1	2	1	0.35	0.05	Slight damage
18	0.03	10.3	0.76	1	0	2	0.36	0.11	Slight damage

Table 4. Neural network training group samples C.

Serial	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Score	Destruction degree
19	0.12	8.3	1.19	1	2	1	0.35	0.46	Medium-sized damage
20	0.05	7.8	4.23	1	2	1	0.34	0.7	Serious damage
21	0.15	4	1.04	0	1	0	0.37	0.37	Low-grade damage
22	-0.04	13.8	2.04	1	2	1	0.35	0.06	Slight damage
23	0.2	9.3	5.62	1	2	1	0.35	0.18	Little serious damage
24	0.07	9.2	5	1	0	2	0.36	0.35	Low-grade damage
25	0.2	9.7	5.6	1	2	1	0.35	0.15	Slight damage
26	0.07	8.3	0.88	1	2	1	0.34	0.65	Little serious damage
27	0.08	7.6	0.57	1	2	1	0.35	0.5	Medium-sized damage

Table 5. Neural network training group samples D.

Serial	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Score	Destruction degree
28	0.01	13.4	1.27	1	2	1	0.34	0.42	Medium-sized damage
29	-0.01	5.3	2.06	1	2	1	0.35	0	No damage
30	0.11	10.2	0.99	1	1	0	0.36	0.06	Low-grade damage
31	-0.07	15.4	3.8	1	2	1	0.34	0.07	Slight damage
32	0.08	5.6	4.51	1	2	1	0.35	0.55	Medium-sized damage
33	-0.33	8.5	2.27	1	0	2	0.36	0	No damage
34	0.01	5.2	0.71	1	2	1	0.35	0.57	Medium-sized damage
35	0.02	19	0.51	1	0	2	0.36	0.05	Slight damage
36	0.03	10.3	4.48	1	2	1	0.35	0.26	Low-grade damage

Table 6. Neural network test samples.

Serial	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Score	Destruction degree
1	0.25	3	5.54	1	2	1	0.35	0.93	Serious damage
2	0.12	7.5	0.73	1	2	1	0.35	0.44	Medium-sized damage
3	-0.18	15	2.76	0	1	0	0.38	0	No damage
4	-0.003	9.6	5.12	1	2	1	0.35	0.35	Low-grade damage

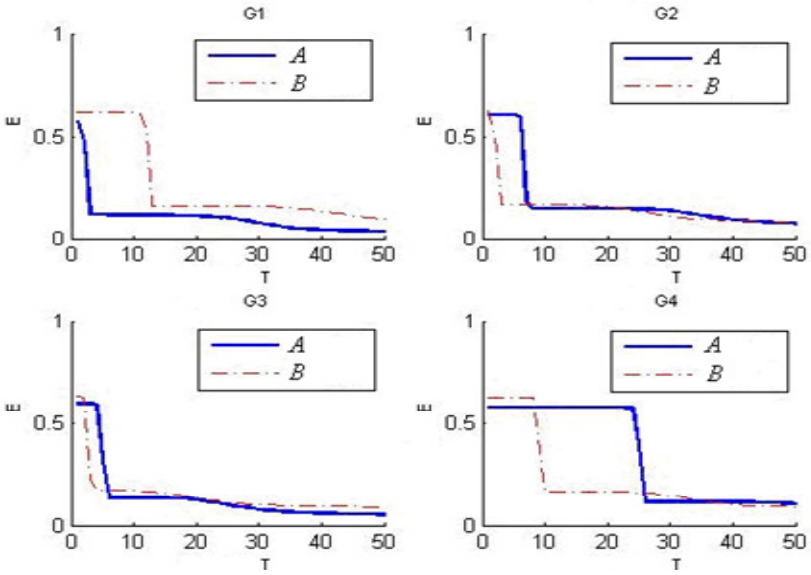

Figure 1. NN A_1 training result.

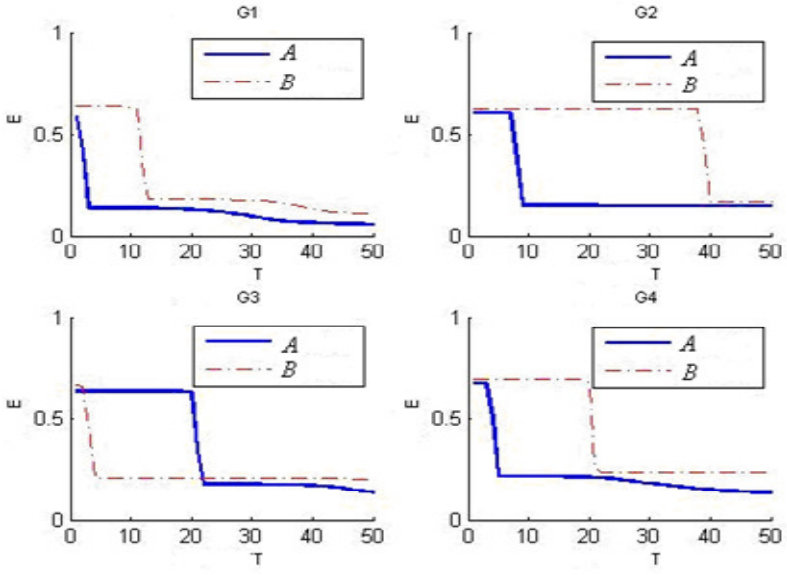

Figure 2. NN A_2 training result.

G are trained respectively by the neural network A_1, A_2, A_3. The training process is shown in Figures 1 to 3. The dashed lines in the figures denote the result of neural network iterating 50 times which has not been optimized by PSO algorithm. The real lines in the figures denote the

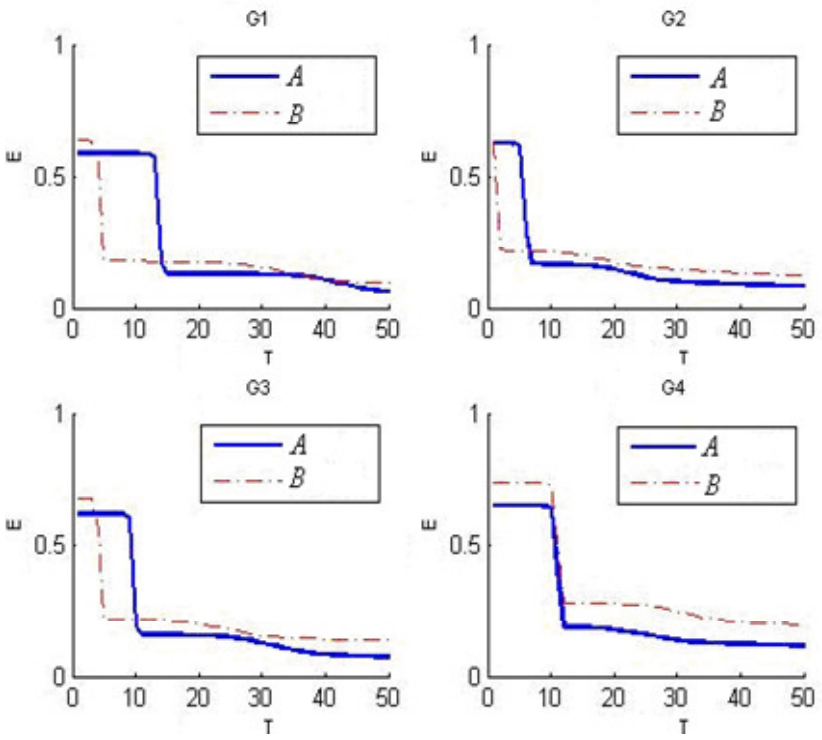

Figure 3. NN A_3 training result.

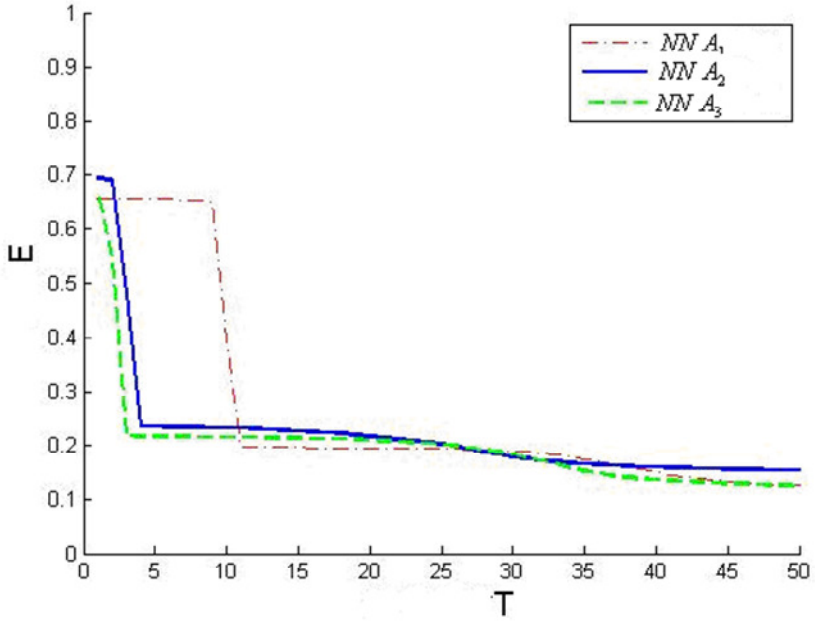

Figure 4.Overall sample collection training result.

result of neural network iterating 50 times which has been optimized by PSO algorithm

From Figures 1 to 3, on one hand, PSO-based neural network demonstrates obvious performance during the training. On the other hand, for different sample collections, different neural networks demonstrate different performances; “credibility” is not the same.

Figure 4 is the iterative process of three kinds of PSO-

Table 7. Calculation result of average error.

Variable	E_1	E_2	E_3	E_4	E
A_1	0.05	0.06	0.09	0.13	0.03
A_2	0.02	0.06	0.05	0.03	0.05
A_3	0.04	0.08	0.08	0.11	0.04

Table 8. Intuitionistic fuzzy decision-making matrix R.

Variable	G_1	G_2	G_3	G_4
A_1	(0.17,0.45)	(0.34,0.30)	(0.33,0.41)	(0.32,0.48)
A_2	(0.69,0.18)	(0.34,0.30)	(0.34,0.23)	(0.10,0.11)
A_3	(0.14,0.37)	(0.32,0.40)	(0.33,0.36)	(0.58,0.41)

Table 9. Intuitionistic judgment matrix B.

Variable	A_1	A_2	A_3
A_1	(0.5,0.5)	(0.3,0.7)	(0.4,0.6)
A_2	(0.7,0.3)	(0.5,0.5)	(0.8,0.2)
A_3	(0.6,0.4)	(0.2,0.8)	(0.5,0.5)

Table 10. Score matrix S.

Variable	G_1	G_2	G_3	G_4
A_1	-0.28	0.04	-0.07	-0.16
A_2	0.51	0.04	0.11	-0.01
A_3	-0.23	-0.08	-0.03	0.17

based neural networks to train overall sample G. It shows different errors. The grouped sample collection G1, G2, G3 and G4 and overall sample collection G are trained respectively ten times by the neural network A_1, A_2, A_3. We obtain the average error as the calculation foundation for intuitionistic judgment matrix and intuitionistic fuzzy decision-making matrix. Table 7 shows the calculation result of average error.

Weight determination of different neural networks

We set up the intuitionistic fuzzy decision-making matrix *R* according to the average error E1, E2, E3, E4. It is shown in Table 8. According to the average error E of neural network A1, A2 and A3 to the overall sample collection G, we set up the intuitionistic judgment matrix *B* as Table 9 shows after normalization. Calculate the score matrix S as Table 10 shows according to the intuitionistic fuzzy decision-making matrix R. Matrix S will be turned into the normal matrix as Table 11 shows. We can get the optimum attribute weight vector of G1, G2, G3 and G4 according to the linear programming model.

$w^* = (0.3262, 0.1907, 0.3805, 0.1125)$. Then we can get the synthetic score of neural network A1, A2, A3 according to formula (10) and (11).

$$z_1 = [0.4957, 0.6768]$$

$$z_2 = [0.5815, 0.8153]$$

$$z_3 = [0.6198, 0.7677]$$

Then we can set up the possibility degree matrix P as Table 12 shows according to the comparison between the synthetic scores. scores. We can get the weight vector of A1, A2, and A3 utilizing formula (14).

$$\omega = (0.2684, 0.4458, 0.3149)$$

Assessment of test samples

We can get the weighted collection using the weight vector to the assessment value of three kinds of neural networks. The assessment value obtained finally using

Table 11. Normal matrix $\overline{S}$.

	G_1	G_2	G_3	G_4
A_1	0	1	0.75	0.375
A_2	1	0.096	0.231	0
A_3	0	0.375	0.5	1

Table 12. Possibility degree matrix P.

Variable	A_1	A_2	A_3
A_1	0.5	0.1825	0.5155
A_2	0.8576	0.5	0.7124
A_3	0.5792	0.3022	0.5

Table 13. Assessment result of test samples.

Variable	Test samples 1			Test samples 2		
	Assessing value	Actual value	Error	Assessing value	Actual value	Error
A_1	0.89	0.93	0.04	0.41	0.44	0.03
A_2	0.95	0.93	0.02	0.39	0.44	0.05
A_3	0.91	0.93	0.02	0.47	0.44	0.03
A_4	0.94	0.93	0.01	0.43	0.44	0.01
	Test samples 3			Test samples 4		
A_1	0	0	0	0.40	0.35	0.05
A_2	0.07	0	0.07	0.37	0.35	0.02
A_3	0.15	0	0.15	0.33	0.35	0.02
A_4	0.06	0	0.06	0.36	0.35	0.01

fuzzy artificial neural network fusion algorithm is the assessment of test samples. The assessment results and analyses are shown in Table 13. We can arrive at a conclusion that when we use the fusion algorithm to assess, the error is lower than the single neural network except the assessment for test samples
3. The effectiveness and reasonableness of fusion algorithm are proved.

CONCLUSIONS

The paper combines the intuitionistic fuzzy sets theory and PSO algorithm with the neural network, and applies it to the comprehensive evaluation of target damage effect in the battlefield. Through the instantiate analysis, we have verified its validity and rationality.

REFERENCES

Chang YP, Koh CN (2009). A PSO method with nonlinear time-varying evolution based on neural network for design of optimal harmonic filters. Expert Syst. Appl., 36(3): 6809-6816.
Chau KW (2007). Application of a PSO-based neural network in analysis of outcomes of construction claims. Autom. Construct., 16(5): 642-646.
Chen ZH (2010). PSO-based back-propagation artificial neural network for product and mold cost estimation of plastic injection molding. Comput. Industr. Eng., 58(4): 625-637.
Fan WM (2004). Study on the neural network based on the genetic algorithm. J. Taiyuan Univ., 3(4): 14-17.
Hong DH (2000). Multi criteria fuzzy decision making problems based on vague set theory. Fuzzy Sets and Systems, 114(1): 103-113.
Jing YW, Ren T (2006). Neural network training using PSO algorithm in ATM traffic control. Intelligent Control and Automation 344: 341-350.
Liang XY, Yang Z (2009). Research on activated carbon super capacitors electrochemical properties based on improved PSO-BP neural network. CMC-Computers Materials and Continua, 13(2): 135-151.
Mikhailov L (2005). Deriving priorities from fuzzy pair wise comparison judgments. Fuzzy Sets and Systems, 152(3): 475-498.
Mu YQ, Sheng AD (2009). Evolutionary diagonal recurrent neural network with improved hybrid EP-PSO algorithm and its identification application.International J. Innovative Comput. Inf., 5(6): 1615-1624.
Song Y, Chen ZQ (2007). New chaotic PSO-based neural network predictive control for nonlinear process.IEEE Transactions on Neural Networks, 18(2): 595-600.
Wang FQ, Gao Y (2001). Optimization of the neural network based on the genetic algorithm. J. Yanshan Univ., 25(3): 234-238.
Wang P, Huang ZY (2008). Mechanical property prediction of strip

model based on PSO-BP neural network. J. Iron Steel Res. Int., 15(3): 87-91.
Xu ZS (2004). Uncertain multiple attribute decision making: methods and applications. Beijing: Tsinghua University Press.
Xu ZS (2007). An iterative method for fuzzy multiple attributes group decision making. Inf. Sci., 177(1): 248-263.
Zhou JL, Duan ZC (2006). PSO-based neural network optimization and its utilization in a boring machine. J. Mater. Processing Technol., 178: 1-3.