Expert Systems with Applications 36 (2009) 5402–5410

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Pattern recognition using neural-fuzzy networks based on improved particle swam optimization

Cheng-Jian Lin [a,*], Jun-Guo Wang [b], Chi-Yung Lee [c]

[a] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung County, Taiwan 411, R.O.C
[b] Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung County, Taiwan 413, R.O.C
[c] Department of Computer Science and Information Engineering, Nankai Institute of Technology, Nantou, Taiwan 542, R.O.C

ARTICLE INFO

Keywords:
Neural-fuzzy network
Improvement evolutionary direction operator (IEDO)
Human body classification
Skin color detection

ABSTRACT

This paper introduces a recurrent neural-fuzzy network (RNFN) based on improved particle swarm optimization (IPSO) for pattern recognition applications. The proposed IPSO method consists of the modified evolutionary direction operator (MEDO) and the traditional PSO. A novel MEDO combining the evolutionary direction operator (EDO) and the migration operation is also proposed. Hence, the proposed IPSO method can improve the ability of searching global solution. Experimental results have shown that the proposed IPSO method has a better performance than the traditional PSO in the human body classification and the skin color detection.

1. Introduction

Pattern recognition, one of the core techniques in computer applications, how to recognize pattern is very important for information learning. In the past, traditional pattern recognition problems usually use statistic methods (Abd-Almageed & Smith, 2002; Bicego et al., 2001) to analyze. However, the input pattern may be rotated or scale-changed to its standard form and the patterns can be corrupted by noise. The ability of the above-mentioned methods is still limited and unable to achieve better efficiency in recognition. Therefore, there are many correlation research looked for new methods overcome these problems.

In recent years, neural-fuzzy networks (Jang, 1993; Lin & Lee, 1996) have become a popular research topic (Castellano, Fanelli, & Mencar, 2002). They are widely applied in fields such as pattern recognition (Chiang & Hao, 2003), time-series prediction (Kasabov & Song, 2002) and control problem (Lin & Chen, 2003). Developing an intelligent visual surveillance helps replace traditional passive video surveillance and skin color detection. The reason is that neural-fuzzy networks combine the semantic transparency of rule-based fuzzy systems with the learning capability of neural networks. The main advantage of the neural-fuzzy network is that the black box nature of the neural-network paradigm is resolved, as the connectionist structure of a neural-fuzzy network essentially defines the IF–THEN rules. Moreover, a neural-fuzzy network can adjust the parameter of the fuzzy rules using neural-network-based learning algorithms.

The training of the parameters is the major problem in designing a neural-fuzzy system. To solve this problem, back-propagation (BP) (Lee & Teng, 2000; Lin & Chin, 2004; Lin, Shieh, Teng, & Shieh, 2005) training is widely used. It is a powerful training technique that can be applied to networks with a forward structure. Since the steepest descent technique is used in BP training to minimize the error function, the algorithm may reach the local minima very fast and never find the global solution. In addition, the performance of BP training depends on the initial values of the system parameters, and for different network topologies one has to derive new mathematical expressions for each network layer.

The advent of evolutionary computation has inspired new designs and models, such as the optimal design of neural networks and fuzzy models, for optimal problem solving. In contrast to traditional computation systems, which may be good at accurate and exact computation but have brittle operations (i.e. for different topologies one has to derive new mathematical expressions), evolutionary computation provides a more robust and efficient approach for solving complex real-world problems (Back & Schwefel, 1993; Fogel, 1995; Yao, 1999). Many evolutionary algorithms, such as genetic algorithms (GA) (Goldberg, 1989), genetic programming (Koza, 1992), evolutionary programming (Fogel, 1994), particle swarm optimization (PSO) (Eberhart & Kennedy, 1995a; Eberhart & Kennedy, 1995b) and evolution strategies (Rechenberg, 1994), have been proposed. Since they are heuristic and stochastic, they are less likely to get stuck at the local minimum, and they are based on populations made up of individuals with specific behaviors similar to certain biological phenomena. These common characteristics have led to the development of evolutionary computation as an increasingly important field.

* Corresponding author.
E-mail address: cjlin@cyut.edu.tw (C.-J. Lin).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.06.110

A GA (Karr et al., 1993) is first used to design the membership function (IF–THEN rules) of a fuzzy controller, with the fuzzy rule set assigned in advance. GAs are efficient at exploring the entire search space, but are relatively poor in finding the precise local optimal solution in the region in which the algorithm converges. Therefore, Yamamoto and Inoue (1995) uses the traditional evolutionary direction operator (EDO) that selects good target points, and other points will move toward these good target points method with enhances GAs to search for the optimal solution. Chiang (2005) proposed an improved genetic algorithm that uses improved evolutionary direction operator (IEDO) to enhance GA for power economic dispatch of units with vale-point effects and multiple fuels. Therefore, we proposed a modified evolutionary direction operator (MEDO) enhance algorithms to search for the optimal solution capability. Recent a new optimization algorithm, called particle swarm optimization (PSO) (Eberhart & Kennedy, 1995b), was proposed. The PSO possesses obtain the optimal solution in search space. However, the PSO may easily get trapped in a local optimal solution when solving complex problems. In this paper, we will enhance the traditional PSO to enable it to obtain optimal solution capability. Therefore, a new algorithm, called the improved particle swarm optimization (IPSO), is proposed.

In this paper, we propose a recurrent neural-fuzzy network (RNFN) based on improved particle swarm optimization (IPSO) for solving pattern recognition problems. The proposed IPSO method consists of the modified evolutionary direction operator (MEDO) and the traditional PSO. The proposed MEDO improves the global search solution capability of the traditional PSO. Experimental results have shown that the proposed IPSO method performs better than the traditional PSO methods.

This paper is organized as follows. Section 2 describes the structure of the recurrent neural-fuzzy network. The proposed improved particle swarm optimization (IPSO) is presented in Section 3. Section 4 describes the experimental results in posture classification and skin color detection. Finally, conclusions are given in the last section.

2. A recurrent neural-fuzzy network

In this section, we will introduce the recurrent neural-fuzzy network (RNFN). The RNFN model was proposed by Lin and Chin (2004). In Lin and Chin (2004), each fuzzy rule corresponding to a wavelet neural network (WNN) consists of single-scaling wavelets. The nonorthogonal and compactly supported functions are adopted as wavelet neural-network bases.

The structure of the RNFN is shown Fig. 1 where the functions of the node in each layer are described as follows:

Layer 1: Each node in this layer is an input node. These nodes only pass the input signal to the next layer.

$$O_i^{(1)} = x_i^{(1)}. \tag{1}$$

Layer 2: Each node in this layer acts as a membership function representing the term of the respective input-linguistic variables; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is determined in this layer. The Gaussian function given below is adopted as the membership function:

$$O_i^{(2)} = \exp\left(-\frac{(I_{ij}^{(2)} - mij)^2}{\sigma_{ij}^2}\right), \tag{2}$$

where m_{ij} and σ_{ij} are the mean and standard deviation, respectively. Additionally, the input of this layer for the discrete time scan be denoted by

$$I_{ij}^{(2)} = O_i^{(1)} + O_{ij}^{(f)}(t), O_{ij}^{(f)}(t) = O_{ij}^{(2)}(t-1) \cdot \theta_{ij}, \tag{4}$$

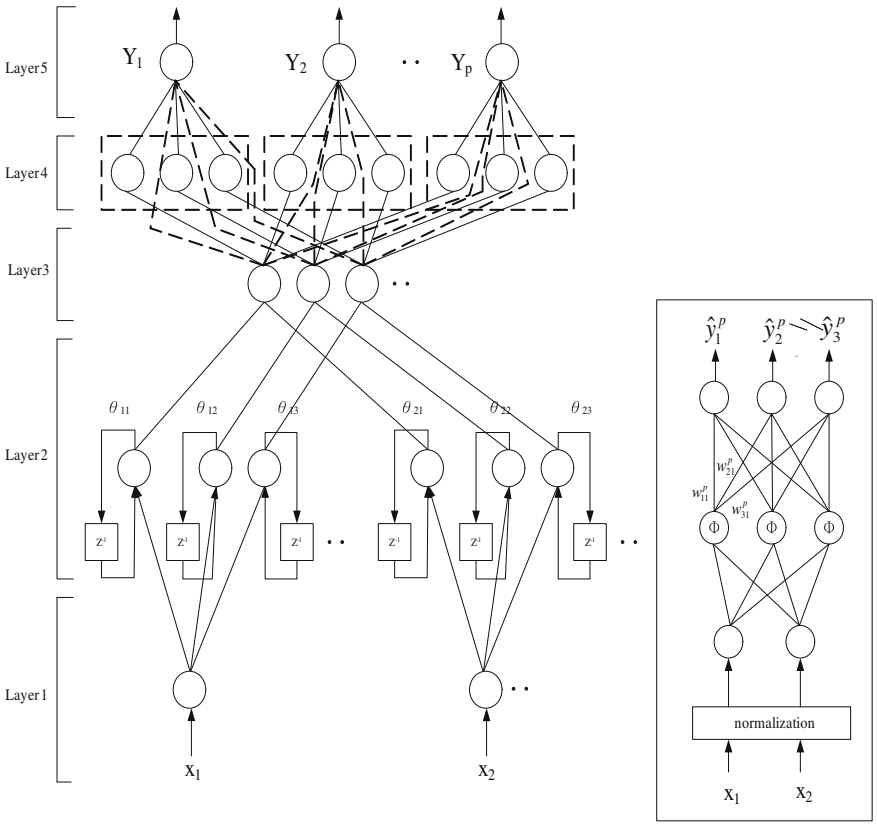

Fig. 1. Schematic diagram of the RNFN model.

where θ_{ij} is the feedback weight. Clearly, the input of this layer contains the memory terms $O_{ij}^{(2)}(t-1)$, which store the past information of the network.

Layer 3: Each node in this layer is a rule node representing the precondition part of one fuzzy logic rule. Therefore, each node of this layer is denoted by Π, which multiplies the incoming signals from layer 2 and outputs the product result, i.e., the firing strength of a rule. For the jth rule node

$$O_j^{(3)} = \prod_{i=1}^{n} O_i^{(2)} = \prod_{i=1}^{n} \exp\left(-\frac{(I_{ij}^{(2)} - m_{ij})^2}{\sigma_{ij}^2}\right), \tag{5}$$

where n is the number of external dimensions.

Layer 4: Nodes in this layer receive the signals, which are $\hat{y}_j$ from the output of the wavelet neural-network model and $O_j^{(3)}$ from the output of layer 3. The mathematical function of each node j is

$$O_j^{(4)} = \hat{y}_j^p \cdot O_j^{(3)}, \tag{6}$$

$$\hat{y}_j^p = \sum_{k=1}^{M} w_{jk}^p \varphi_{a.b}, \tag{7}$$

where $\hat{y}_j^p$ denotes the local output of the WNN for the output and the jth rule, and the link weight w_{jk}^p is the output action strength associated with the pth output

$$\varphi_{a.b} = \frac{\sum_{j=1}^{n} \varphi_{a.b}(x_i)}{|X|}, \tag{8}$$

where $|X|$ is the number of input dimensions. The $\varphi_{a.b}(x_i)$ functions which are used to input vectors to fire up the wavelet interval are then calculated. A value $\varphi_{a.b}$ is obtained as follows:

$$\begin{cases} \varphi(x_i) = \cos(x_i), & -0.5 \leqslant x_i \leqslant 0.5 \\ 0 & \text{otherwise}, \end{cases} \qquad \varphi_{a.b} = \cos(ax_i - b). \tag{9}$$

Layer 5: The node in this layer computes the output signal Y_p. The output node together with links connected to it acts as a defuzzifier. The mathematical function is

$$\begin{aligned} Y_p &= \frac{\sum_{j=1}^{M} O_j^{(4)}}{\sum_{j=1}^{M} O_j^{(3)}} = \frac{\sum_{j=1}^{M} \hat{y}_j^p O_j^{(3)}}{\sum_{j=1}^{M} O_j^{(3)}} \\ &= \frac{\sum_{j=1}^{M} (w_1^j \phi_{1.1} + w_2^j \phi_{2.1} + \cdots + w_k^j \phi_{m.m}) O_j^{(3)}}{\sum_{j=1}^{M} O_j^{(3)}}. \end{aligned} \tag{10}$$

Details of the structure of the RNFN model can be found in Lin and Chin (2004).

3. An improved particle swarm optimization

Particle swarm optimization (PSO) was originally introduced by Kennedy and Eberhart in 1995 for the study of social and cognitive behavior (Eberhart & Kennedy, 1995a, 1995b). The idea originated in studies on the synchronous flocking of birds and schooling of fish. The PSO algorithm has come to be widely used as a problem solving method in engineering and computer science. This algorithm has several highly desirable attributes, including a basic algorithm that is very easy to understand and implement. It is similar in some ways to evolutionary algorithms, but requires less computational bookkeeping and generally fewer lines of code.

In the PSO, the trajectory of each individual in the search space is adjusted by dynamically altering the velocity of each particle, according to its own flight experience and the flight experience of the other particles in the search space. The position vector and the velocity vector of the ith particle in the N-dimensional search space can be represented by $X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{id})$ and $V_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{id})$, respectively. According to an user-defined fitness function, suppose that the best position of each particle (which corresponds to the best fitness value obtained by that particle at time) is $P_i = (p_{i1}, p_{i2}, p_{i3}, \ldots, p_{id})$, and the fittest particle found so far is $P_g = (p_{g1}, p_{g2}, p_{g3}, \ldots, p_{gd})$. Then the new velocities and the positions of the particles for the next fitness evaluation are calculated using the following two equations:

$$v_{vid}^{k+1} = \omega \times v_{vid}^k + c_1 \times \text{rand}(\cdot) \times (P_{id} - x_{id}^k) + c_2 \times \text{rand}(\cdot) \times (P_{gd} - x_{id}^k) \tag{11}$$

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1}, \tag{12}$$

where w, c_1 and c_2 are called the coefficients of the inertia term, the cognitive term and the society term, respectively. Rand yields uniformly distributed random numbers in [0, 1].

The first part of Eq. (11) represents the previous velocity, which provides the necessary momentum for particles to roam across the search space. The second part of Eq. (11), known as the "cognitive" component, represents the personal thinking of each particle. The cognitive component encourages the particles to move toward their own best positions found so far. The third part of Eq. (11) is known as the "social" component, which represents the collaborative effect of the particles, in finding the global optimal solution. The social component always pulls the particles toward the global best particle found so far.

The traditional evolutionary direction operator (EDO) (Juang & Lin, 1998) was first used for GAs. The EDO selects good target points (i.e., particles), and other points will move toward these good target points. Because the GA is a random search method, it cannot obtain optimal solutions efficiently and quickly. The EDO method can enhance GAs to search for the optimal solution. The main shortcoming of the EDO is that the new particle created from three arbitrary particles in each generation cannot be certain to have good evolutionary direction. For this reason, the evolutionary direction may not be toward a better direction. Therefore, we propose a modified evolutionary direction operator (MEDO) to improve this shortcoming of the traditional EDO. We use the MEDO to enhance the capability of the traditional PSO to find the optimal solution. The new algorithm is called the improved particle swarm optimization (IPSO). In this study, a Gaussian membership function is adopted with variables that represent the mean and deviation of the membership function. The fuzzy rule given by Eq. (5), where m_{ij} and σ_{ij} are the mean and deviation of a Gaussian membership function, respectively, and w_{ij} represents the corresponding link weight of the consequent part that is connected to the jth rule node. In this study, a real number represents the position of each particle. Fig. 2 shows the flowchart of the proposed IPSO learning process. The whole learning process is described step-by-step as follows.

3.1. Individual initialization

The individual initialization step sets the initial values for every particle. The individual initialization step sets the initial values for every particle. Each particle includes the mean, deviation and weight variables of the RNFN, and their values are generated randomly.

3.2. Evaluate fitness

The evaluation step evaluates each particle in a swarm. The fitness function is defined as follows:

$$f_i = 1/Y, \tag{13}$$

$$Y = \sqrt{\frac{1}{N}\sum_{p=1}^{N}(y_p - \bar{y}_p)}, \quad \text{for } p = 1, 2, \ldots, N, \tag{14}$$

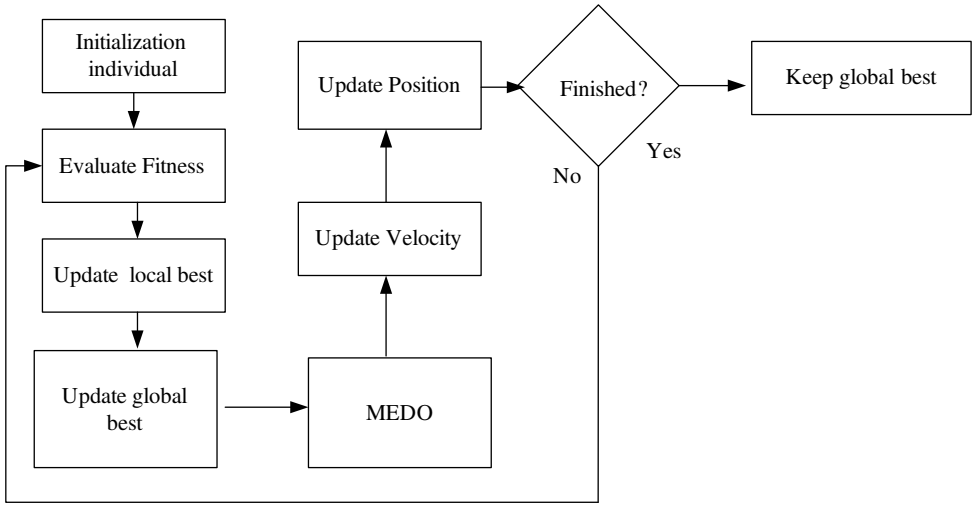

Fig. 2. The flowchart of the proposed IPSO learning process.

where N represents the number of input data; y and $\bar{y}$ represent the model output and the desired output, respectively.

3.3. Update the local best and the global best

In this step, we update the local best and the global best. The updating process of the local best and the global best is as follows:

1. If the fitness value of the ith particle is higher than that of the ith local best, then the ith local best will be replaced with the ith particle.
2. If the fitness value of the ith particle is higher than that of the global best, then the global best will be also replaced with the ith particle.

3.4. Modified evolutionary direction operator

With the MEDO, we choose the three best solutions in each generation to perform the evolutionary direction operation. The new solution is superior to the original best solution.

After a generation of learning, the three best particles are obtained. These three best particles are ordered according to their fitness and are called the "low," "medium," and "high" particles. Three inputs (preferred) and the output (created) particles are denoted as follows: Input particles:

"low" particle, $C_l = (C_{l1}, C_{l2}, C_{l3}, \ldots, C_{ld})$, with fitness F_l.
"medium" particle, $C_m = (C_{m1}, C_{m2}, C_{m3}, \ldots, C_{md})$, with fitness F_m.
"high" particle, $C_h = (C_{h1}, C_{h2}, C_{h3}, \ldots, C_{hd})$, with fitness F_h.
Output particle, $C_o = (C_{o1}, C_{o2}, C_{o3}, \ldots, C_{od})$, with fitness F_o.

First, the "low" particle is updated using a migration operation to generate a new "low" particle. Next, the "medium" particle and the "high" particle will be set as the moving target direction of the new "low" particle. That is, the new "low" particle will be updated again toward the "medium" and "high" particles. This process improves the capability of finding the global solution. Fig. 3 shows the flowchart of the proposed MEDO. The detailed MEDO is described as follows:

Step 1: Set the magnitudes of the two evolution directions to 1 (i.e., $D = 1$, $D = 1$). Then set the initial index of the MEDO to 1 (i.e., $T_s = 1$), the number of the MEDO loop to N_L, and the three particles with the best fitness values from the local best swarm to C_h, C_m, and C_l.

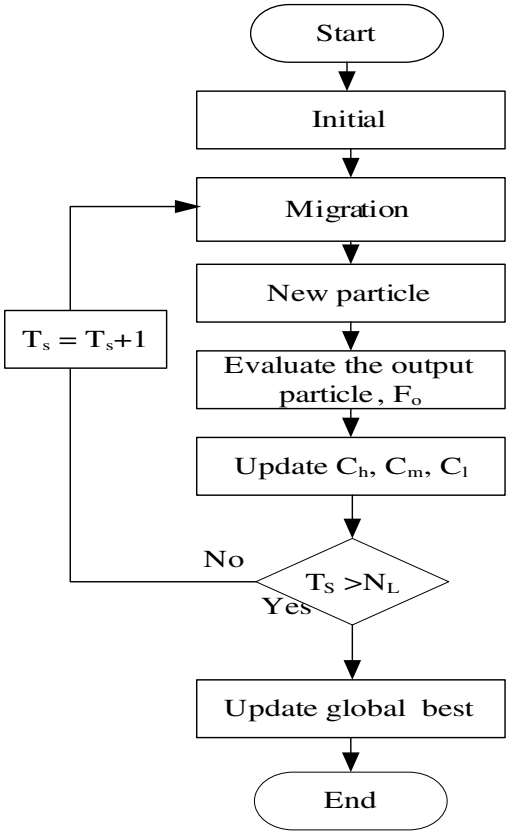

Fig. 3. Flowchart of the proposed MEDO.

Step 2: The migration operation (Chiang, 2005) in the MEDO is used to regenerate a newly diverse population, which prevents individuals from gradual clustering. Thus, the migration operation greatly increases the amount of search space explored for a small swarm. The migrant individuals are generated based on the best individual, $X_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{id})$, by non-uniform random selection. We use Eq. (15) to update the low particle (C_l) and the medium particle (C_m).

$$x_{id} = \begin{cases} x_{id} + \rho(x_{id}^{L} - x_{id}), & \text{if } r_1 < \frac{x_{id} - x_{id}^{L}}{x_{id}^{L} - x_{id}^{U}} \\ x_{id} + \rho(x_{id}^{U} - x_{id}), & \text{otherwise}, \end{cases} \quad (15)$$

where ρ and r_1 are random numbers in the range of [0,1]. If the fitness value is not improved and this optimum value is a local optimum, then we use the migration operation to solve this problem. The migration operation is the migrant x_{id} to other points and this idea can escape the local extreme value trap.

Step 3: Compute C_o using

$$C_{oj} = C_{lj} + D_1 \times (C_{lj} - C_{mj}) + D_2 \times (C_{lj} - C_{hj}). \quad (16)$$

Starting from the base point C and with the use of two difference vectors, $D_1 \times (C_{lj} - C_{mj})$ and $D_2 \times (C_{lj} - C_{pj})$, the next evolutionary direction and the next evolutionary step-size can be determined using this parallelogram. Point C_o can then be created along the evolutionary direction with the evolutionary step-size.

Step 4: Evaluate the new fitness (F_o) of the newly created output particle (C_o).

Step 5: Update the "low" particle (C_l), "medium" particle (C_m), and "high" particle (C_h). The updating process is as follows:

(1) If $F_o > F_h$, then $C_h = C_o$, $C_m = C_h$, and $C_l = C_m$.
(2) Else if $F_o > F_m$ and $F_o < F_h$, then $C_m = C_o$ and $C_l = C_m$.
(3) Else if $F_o > F_l$ and $F_o < F_m$, then $C_l = C_o$.
(4) Else if $F_o = F_l = F_m$, then $C_o = C_o + N_r$ ($N_r \in [0,1]$).
(5) Else if $F_o < F_l$, then $D_1 = D_1 \times -0.5$ and $D_2 = D_2 \times -0.5$.

According to statements (1)–(3) in the process above, we update the "low" particle, the "medium" particle and the "high" particle. In statement (4), when the new particle, the "low" particle, and the "medium" particle all have the same fitness values, the particle will fall into a local optimum. Thus, a random number (N_r) is added to prevent the learning algorithm from falling into a local optimum. If the fitness value of the new particle in statement (5) is not good, we will decrease the moving velocity (i.e., D_1 and D_2) to obtain a good fitness.

Step 6: In this step, we determine whether the MEDO is to be terminated. If the MEDO is terminated, go to step 7. Otherwise, $T_s = T_s + 1$, and go to step 2.

Step 7: In this step, we update the global best. The updating process of the global best is as follows: if the fitness value of the new particle is higher than that of the global best, then the global best will also be replaced with the particle.

3.5. Update velocity and update position

The velocity of all particles along each dimension is updated using Eqs. (11) and (12).

4. Experimental results

In order to demonstrate the performance of the proposed IPSO approach, experiments were conducted, human body classification and skin color detection problems using two different data sets. All the programs were developed using Borland C++ Builder 6.0, and each problem was simulated 10 times on a Pentium IV 3.2 GHz desktop computer.

4.1. Example 1: Human body classification problem

We adopt length–width ratio to compute body's silhouette and get a posture feature. We derive from vertical and horizontal projection histograms of segmented people. To continue, other body posture features are extracted from histograms of body silhouette horizontal and vertical projections. However, relies on the body silhouette size and position. If silhouette size differs as the distance from people to camera varies and the same silhouette size, the projection histogram differs as a person's image position differs. Therefore, we exploit discrete Fourier transform (DFT) method to perform in this paper. Because, DFT is performed on a vertical and horizontal projection histogram of segmented people to solve the scaling and shifting problem. So, the proposed DFT coefficients are used as features and we extracted important 20 significant DFT coefficients features of different postures.

For human body classification classifier training, the initial parameters before training are given in Table 1 and we adopt 80 training data, with 20 training data for each of the postures. Only four rules are generated after learning (standing, bending, sitting, and lying). For training, 320 train images collected from different posture views are used, with 80 images for each of the four postures. For testing, 400 test images collected from different posture views are used, with 100 images for each of the four postures. The output of the best performance in the IPSO-RNFN is shown in Fig. 4. The x-axis represents the continue posture feature patterns and the y-axis represents the four classes of the human postures – stand, crow, sit and lying. In Fig. 4, the label "o" means that the pattern is classified as the correct class while the label "x" means that the pattern is classified as the incorrect class. The recognition results for each posture are shown in Table 2 where the average recognition rate of our method is 98.25%. In addition, though the body-segmented result may be imperfect, the proposed classification system still achieves a high classification rate, showing the robustness of the system.

In this example, we compared the performance of the proposed IPSO method with the traditional PSO (Eberhart & Kennedy, 1995b) method. The learning curves are shown in Fig. 5. In Fig. 5, we find that the performance of the proposed IPSO method is superior to the PSO method. The comparison items the testing accuracy rates (e.g. best, worst and average situations), training and testing errors, and cost. The comparison results with various existing models are tabulated in Table 3.

4.2. Example 2: Skin color detection problem

Current human recognition methods, such as fingerprinting, retinal scanning, and voice detection have become mature technologies. The aforementioned detection processes generally require a cooperative subject, views from certain aspects, and physical contact with close subject proximity. Often people feel that these detection systems violate their privacy. Skin color detection is the only non-contact detection method. In skin color detection, users pass through detection areas, with a computer system scanning the face in a non-intrusive manner. Though the skin color detection plays an important role in applications such as face recognition and face image database management, being able to detect faces is a crucial step in identification applications.

Three input dimensions (Y, Cb, and Cr) were used in this experiment. We chose 6000 training and testing data. We used the CIT database to produce both the training data and the testing data. The 6000 skin and non-skin pixel training data in the color images were randomly chosen. The testing data were also chosen

Table 1
The initial parameters before training

Generations	1000	V_{max}	2
Number of rules	5	c_1 and c_2	2
Particles	100	ω	0.3

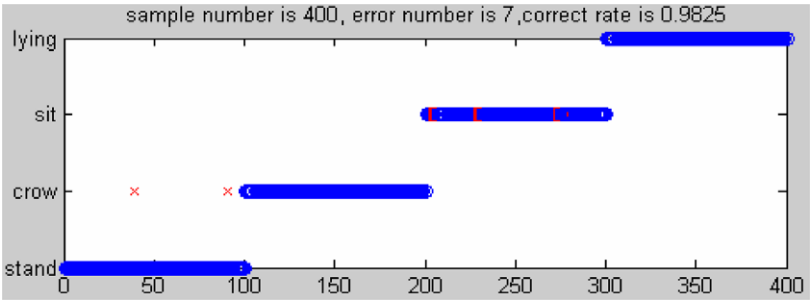

Fig. 4. The results of correct and incorrect posture classification.

Table 2
The posture classification result by RNFN, where the columns indicate the percentage of the true posture and rows indicate the classification

	Standing	Bending	Sitting	Lying	Average
Standing	98	0	0	0	
Crowing	2	100	5	0	
Sitting	0	0	95	0	
Lying	0	0	0	100	
Recognition rate (%)	98	100	95	100	98.25

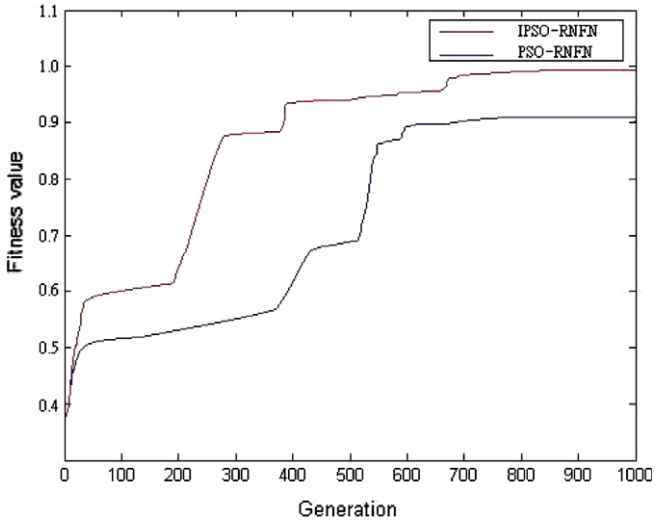

Fig. 5. The learning curves of the two methods using the IPSO-RNFN and PSO-RNFN.

Table 3
Performance comparison with various existing models from 400 test Images

	IPSO	PSO
Testing	400	400
Best/worst accuracy rate (testing)	98.25%/90.25%	90.25%/77.5%
Average testing accuracy rate (%)	94.5	85.75
Generations	1000	1000

randomly. We set four rules constituting a recurrent neural-fuzzy network. We observe that 100 generations have the best performance. Finally, we compared our method with other methods.

We used the California Institute of Technology (CIT) face database on http://www.vision.caltech.edu/Image_Datasets/faces/.

Table 4
The initial parameters before training

Generations	100	V_{max}	2
Number of rules	5	c_1 and c_2	2
Particles	100	ω	0.4

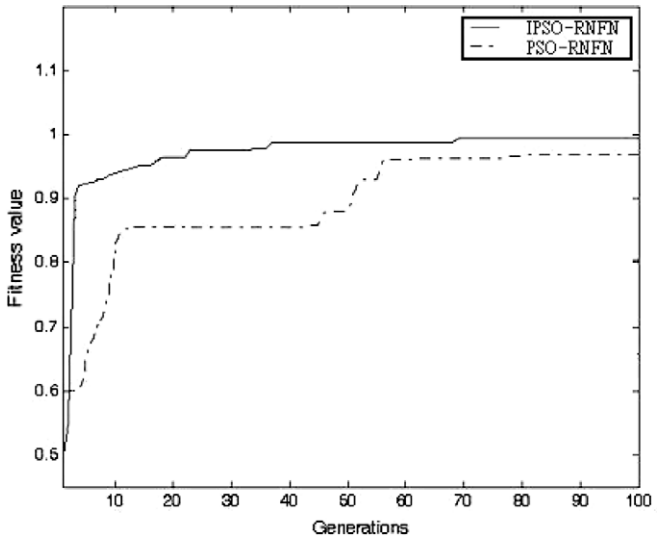

Fig. 6. The learning curves of the IPSO and the PSO methods for the CIT database.

Table 5
Performance comparison with various models from the CIT database

	IPSO	PSO
Training data	6000	6000
Best/worst accuracy rate (testing)	99.33%/88.78%	94.16%/85.9%
Average testing accuracy rate (%)	94.13	90.19
Generations	100	100

The database has 450 color images, the size of each being 320*240 pixels, and contains 27 different people and a (W) variety of lighting, backgrounds, and facial expressions.

Three input dimensions (Y, Cb, and Cr) were used in this experiment. We chose 6000 training and testing data. We used the CIT database to produce both the training data and the testing data. The 6000 skin and non-skin pixel training data in the color images were randomly chosen. The testing data were also chosen

randomly. We set five fuzzy rules constituting a recurrent neural-fuzzy network and compared our method with other methods.

The initial parameters before training are given in Table 4. The learning curves are shown in Fig. 6. In this figure, we find that the performance of the proposed IPSO method is superior to the traditional PSO method. In this example, as with example 1, the performance of the IPSO method is also compared with the traditional PSO. The comparison items include the best testing accuracy rates, the worst testing accuracy rates, and the average testing accuracy rates. The comparison results with various models are tabulated in Table 5.

The California Institute of Technology face (CIT) database consists of complex backgrounds and diverse lighting. The color images from the CIT database are shown in Fig. 7. A well-trained network can generate binary outputs (1/0 for skin/non-skin) to detect a facial region. Fig. 8 shows that our approach accurately determines a facial region. Hence, from the comparison data listed in Table 3, the average of the test accuracy rate is 90.19% for the traditional PSO and 94.13% for the proposed IPSO. This demonstrates that the CIT database is more complex and does not lead to a decrease in the accuracy rate. The proposed IPSO method maintains a superior accuracy rate.

Fig. 7. Original face database from California Institute of Technology (CIT).

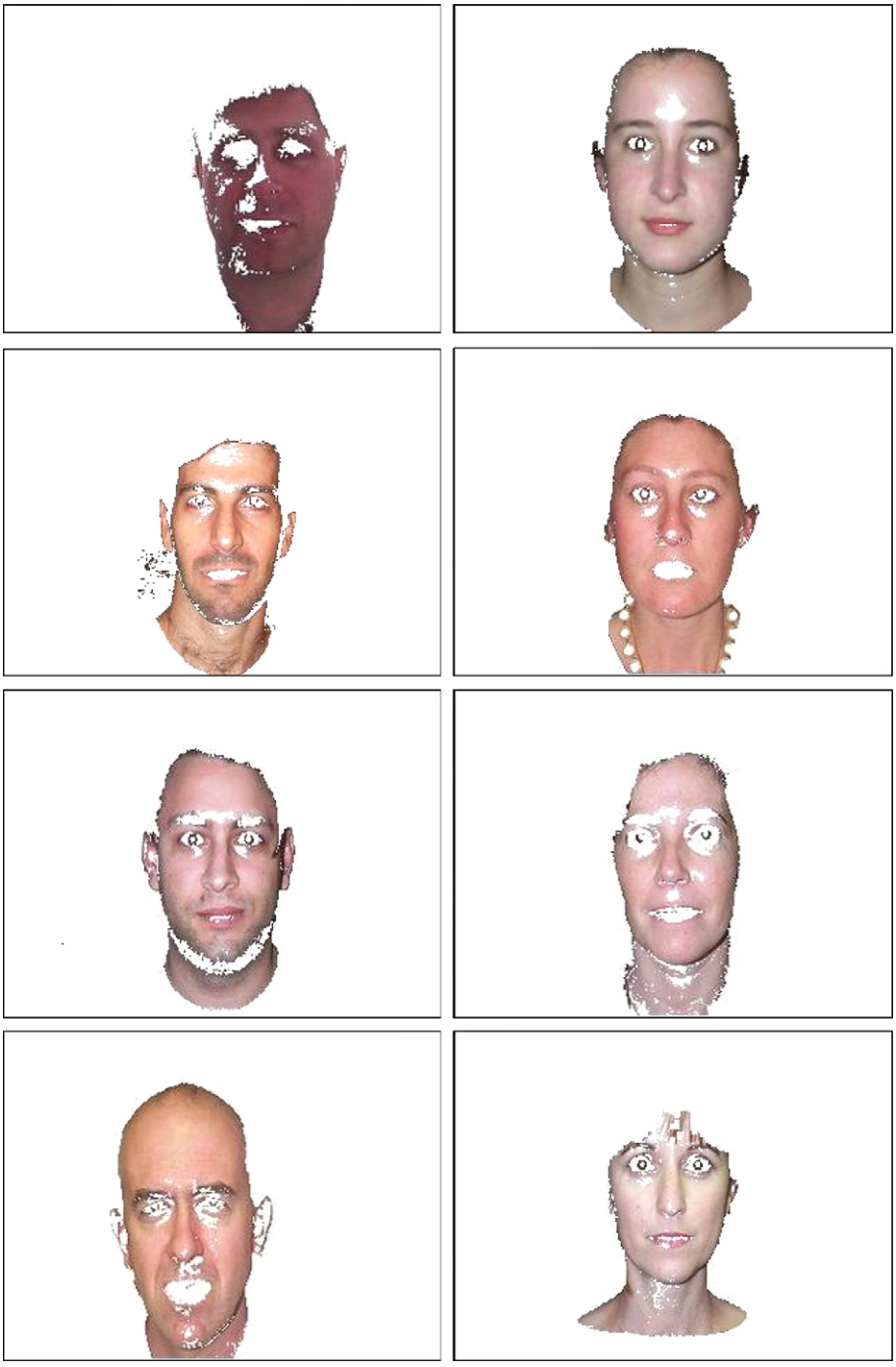

Fig. 8. Results of skin color detection with three dimension input (Y, Cb, and Cr).

5. Conclusion

This paper, proposed a recurrent neural-fuzzy network (RNFN) with an improve particle swarm optimization (IPSO) for solving pattern recognition problems. In RNFN, the consequent part of the rules is a nonlinear function of input-linguistic variables. This study adopts the wavelet neural network to the consequent part of the rules. The local properties of wavelets in the RNFN model enable arbitrary functions to be approximated more effectively. In order to avoid trapping in a local optimal solution and to insure the search capability of a near global optimal solution, mutation plays an important role in IPSO. The experimental results show that our method achieves faster learning and a higher design accuracy rate in the human body classification and skin color detection problems.

References

Abd-Almageed, W., & Smith, C. (2002). Hidden Markov models for silhouette classification. *Proceedings of the world automation congress 2002*, 395–402.

Back, T., & Schwefel, H. P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolution and Computations, 1*(1), 1–23.

Bicego, J. M., & Murino, V. (2001). 2D shape recognition by Hidden Markov models. In Proceedings of the IEEE conference on image analysis and processing (pp. 20–24). Palermo, Italy.

Castellano, G., Fanelli, A. M., & Mencar, C. (2002). A neuro-fuzzy network to generate human-understandable knowledge from data. *Cognitive Systems Research Journal, 3*(2), 125–144.

Chiang, C. L. (2005). Improved genetic algorithm for power economic dispatch of units with vale-point effects and multiple fuels. *IEEE Transactions on Power Systems, 20*(4), 1690–1699.

Chiang, J. H., & Hao, P. Y. (2003). A new kernel-based fuzzy clustering approach: Support vector clustering with cell growing. *IEEE Transactions on Fuzzy Systems, 11*(4), 518–527.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science (MHS'95)* (pp. 39–43).

Eberhart, R., & Kennedy, J. (1995). Particle swarm optimization. In *IEEE international conference on neural networks* (Vol. 4) (pp. 1942–1948).

Fogel, L. J. (1994). Evolutionary programming in perspective: The top–down view. In J. M. Zurada, R. J. Marks, II, & C. Goldberg (Eds.), *Computational intelligence: Imitating life*. Piscataway, NJ: IEEE Press.

Fogel, D. B. (1995). *Evolutionary computation: Toward a new philosophy of machine intelligence*. Piscataway, NJ: IEEE Press.

Goldberg, D. E. (1989). *Genetic algorithms in search optimization and machine learning*. Reading, MA: Addison-Wesley.

Jang, J.-S. R. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on System, Man, and Cybernatics, 23*, 665–685.

Juang, C. F., & Lin, C. T. (1998). An on-line self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems, 6*(1), 12–31.

Karr, C. L., & Gentry, E. J. (1993). Fuzzy control of PH using genetic algorithms. *IEEE Transactions on Fuzzy Systems, 1*(1), 46–53.

Kasabov, N. K., & Song, Q. (2002). DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems, 10*(2), 144–154.

Koza, J. K. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Lee, C. H., & Teng, C. C. (2000). Identification and control of dynamic systems using recurrent fuzzy neural networks. *IEEE Transactions on Fuzzy Systems, 8*, 349–366.

Lin, C. J., & Chen, C. H. (2003). Nonlinear system control using compensatory neuro-fuzzy networks. *IEICE Transactions on Fundamentals, E86-A*(9), 2309–2316.

Lin, C. J., & Chin, C. C. (2004). Prediction and identification using wavelet-based recurrent fuzzy neural networks. *IEEE Transactions on Systems, Man and Cybernetis-Part B: Cybernetics, 34*(5), 2144–2154.

Lin, C. T., & Lee, C. S. G. (1996). *Neural fuzzy systems: A neural-fuzzy synergism to intelligent systems*. Englewood Cliffs, NJ: Prentice-Hall. with disk.

Lin, F. J., Shieh, H. J., Teng, L. T., & Shieh, P. H. (2005). Hybrid controller with recurrent neural network for magnetic levitation system. *IEEE Transactions on Magnetics, 41*(7), 2260–2269.

Rechenberg, I. (1994). Evolution strategy. In J. M. Zurada, R. J. Marks, II, & C. Goldberg (Eds.), *Computational intelligence: Imitating life*. Piscataway, NJ: IEEE Press.

Yamamoto, K., & Inoue, O. (1995). New evolutionary direction operator for genetic algorithms. *AIAA Journal Technical Notes, 33*(10), 1990–1993.

Yao, X. (Ed.). (1999). *Evolutionary computation: Theory and applications*. Singapore: World Scientific.