Journal of Marine Science and Technology

Volume 15 | Issue 1 Article 5

HYBRID RECURSIVE PARTICLE SWARM OPTIMIZATION LEARNING ALGORITHM IN THE DESIGN OF RADIAL BASIS FUNCTION NETWORKS

Ching-Yi Chen
Department of Computer and Communication Engineering, China College of Marine Technology and Commerce, Taipei, Taiwan, R.O.C., f1083@mail.ccmtc.edu.tw

Hsuan-Ming Feng
Department of Management Information, National Kinmen Institute of Technology, Kinmen, Taiwan, R.O.C.

Fun Ye
Department of Electrical Engineering, Tamkang University, Tamsui, Taiwan, R.O.C.

HYBRID RECURSIVE PARTICLE SWARM OPTIMIZATION LEARNING ALGORITHM IN THE DESIGN OF RADIAL BASIS FUNCTION NETWORKS

Ching-Yi Chen*, Hsuan-Ming Feng**, and Fun Ye***

Key words: normalized fuzzy c-means, particle swarm optimization, recursive least-squares, radial basis function networks.

ABSTRACT

In this paper, an innovative hybrid recursive particle swarm optimization (HRPSO) learning algorithm with normalized fuzzy c-mean (NFCM) clustering, particle swarm optimization (PSO) and recursive least-squares (RLS) is proposed to generate radial basis function networks (RBFNs) modeling system with small numbers of descriptive radial basis functions (RBFs) for fast approximating two complex and nonlinear functions. Simulation results demonstrate that the generated NFCM-based learning schemes approach the desired modeling systems within the smaller population sizes.

INTRODUCTION

The RBFNs is a simple but efficient type of feed-forward neural network, which has been designed to approximate nonlinear functions and solve complex problems[1, 2, 4, 6, 15, 16]. In this paper, HRPSO learning algorithm is proposed to select proper parameters of radial basis functions (RBFs) to build the RBFNs modeling system (RBFNMS). The aim of RBFNMS is to efficiently build a common and adaptable mechanism which is applicable to fast approach the desired result and to represent various modeling systems. It is clear that the general initial architecture of the RBFNs is only extracted by examples, this way is seldom approaching to an optimal result. Therefore, it is better to tune associated parameters of RBFNs by the learning algorithm [2, 5, 6]. The determination of the initial RBFs is developed in the process of structure configuration. The purpose of structure configuration is to construct the initial architecture of the RBFNMS which represents the behavior of the given input-output pairs. In the literature on fuzzy clustering, the fuzzy c-means (FCM) clustering algorithm defined by Dunn [8] and generated by Bezdek [3] is the well-known and most powerful method in the application of cluster analysis. Due to the adaptation in the configuring data structure, the normalized FCM (NFCM) clustering algorithm based on the new metric is applied to yield the features over input-output training data. The proposed NFCM algorithm will first cluster the given data set into several groups and then select the cluster centers of the individual group, which will make up initial definition of the RBFNMS, such as the number of cluster centers equal to the number of the RBFs; each location of cluster centers are sequentially assigned to present the initial value of the proposed RBFs and weights between the hidden and output layers. In our research, such collection of available features is becoming useful information to create initial PSO that can speed up convergences of approximation functions. Depending on the support of NFCM to generate initial configuration, PSO and RLS learning algorithm will be applied to optimize the proposed RBFNMS as soon as possible.

Particle swarm optimization (PSO) first introduced by Eberhart and Kennedy in 1995 [12], employs the natural animal's behavior such as bird flocking, fish schooling, and swarm theory to yield the best of the characters among comprehensive old populations. The natural creatures accomplish the heuristic exchange of their own and other creature's best knowledge which has been discovered so far among entire swarm. The proposed PSO algorithm simulates such heuristic learning behavior of natural creatures to discover proper parameters of the discussed system. In PSO learning algorithm, all particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of particles.

Paper Submitted 12/30/05, Accepted 04/20/06. Author for Correspondence: Ching-Yi Chen. E-mail: f1083@mail.ccmtc.edu.tw.

**Department of Computer and Communication Engineering, China College of Marine Technology and Commerce, Taipei, Taiwan, R.O.C.*

***Department of Management Information, National Kinmen Institute of Technology, Kinmen, Taiwan, R.O.C.*

****Department of Electrical Engineering, Tamkang University, Tamsui, Taiwan, R.O.C.*

Then, the position (i. e. solution) of every individual particle will be attracted stochastically toward their associated best positions (i. e. best solutions) in multi-dimensional solution space. In a word, the computation of PSO learning algorithm is dependent on two kinds of important information to achieve its learning goal. The first one is every particle's best experience, which has been better so far. The other one is other neighbor's best experiences (i.e., the best solution found so far). PSO has been demonstrated to resolve some wide range of optimization problems through a metaphor of social interaction [7, 9-11, 14]. However, the training speed of population-based optimization algorithm such as genetic algorithms [19], PSO [12] and so on, are actually dependent on the population sizes.

To speed up the training rate, the collected symbolizations of training data pairs are used to minimize the initial population size (i. e. swarm size) of the PSO. In this article, small available particles which implies information of the RBFNMS is extracted by the NFCM algorithm, then the optimization solution will be contiguously extracted by both PSO and recursive least-squares (RLS) [18, 21] learning algorithm. That is, when the initial structure of RBFNMS is originated, the PSO and RLS learning algorithm will be applied to fast approach desired results. In a word, a small numbers of RBFs and a fewer particles are enough to create the proper RBFNMS. Therefore, the proposed design of the RBFNMS will be efficiently generated within a lower calculation load.

The rest of this paper is organized as the following sections. The constructed RBFNs architecture is presented in Section 2. To generate an appropriate RBFNMS, an efficient HRPSO learning algorithm is discussed in Section 3. In Section 4, two nonlinear approximation problems are utilized to illustrate the effectiveness of the proposed HRPSO learning algorithm. Finally, Section 5 concludes the paper.

RBFNS ARCHITECTURE

In this paper, the normal n-inputs and single-output RBFNs architecture is developed as shown in Figure 1. RBFNs generally consists of three layers, which are input, hidden, and output layers. In this paper, a typical Gaussian function is proposed and described by

$$HE(x, c_i, \delta_i) = \exp\left(-\left(\frac{\left\| x - c_i \right\|^2}{2\delta_i^2}\right)\right) \quad (1)$$

where $\left\| x - c_i \right\|$ is the Euclidean distance between an input vector x and a center c_i, and δ_i represent the deviation of the ith RBFs. In this study, the output of the RBFNs is calculated by the weighed average of the output associated with each hidden unit:

$$y = \frac{\sum_{i=1}^{m} HE_i(x) \cdot w_i}{\sum_{i=1}^{m} HE_i(x)} \quad (2)$$

where w_i is the ith weight between the hidden and output layers, m is the number of hidden nodes, and $HE_i(x)$ is the output of the ith hidden unit.

According to the above respect, parameters of the $HE_i(x)$ $\{c_{i1}, c_{i2}, \ldots, c_{in}, \delta_i\}$ combining with the connection weights (w_i) will determine the RBFNMS. Thus, different parameters set R = $\{c_{i1}, c_{i2}, \ldots, c_{in}, \delta_i, w_i, 1 \le i \le m\}$ determine the different RBFNMS with different performance setting. If there are m initial RBFs needed to be constructed, total m* (n + 2) parameters will be needed to be chosen for designing the best RBFNMS. In this article, the parameter selection problem is formulated as a searching problem and a method based on HRPSO learning algorithm is applied to choosing a proper parameter set R in the solution space. The other detailed explanation will be described in the following section.

PARAMETERS SELECTION BY THE HRPSO LEARNING ALGORITHM

In this section, the HRPSO learning algorithm containing NFCM, PSO, and RLS will be proposed to efficiently generate proper RBFNMS.

At first, the traditional Euclidean norm typed measure is replaced with the proposed robust evaluated distance function, and then the improved FCM learning algorithm, i. e., NFCM, is generated. Finally, input-output training data is partitioned into several categories by the effect of NFCM clustering algorithm. When training samples are available, the NFCM clustering algorithm will consciously derive the information from the analysis of the input-output training data pairs.

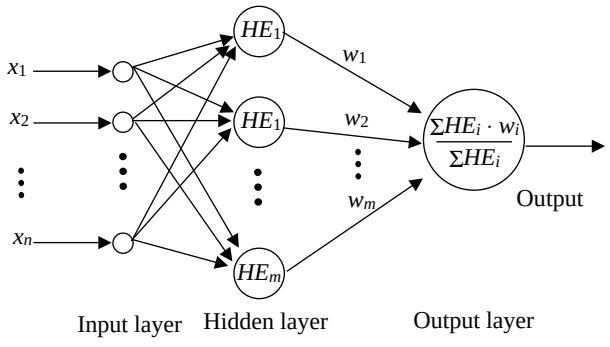


Fig. 1. The proposed architecture of the RBFNs.

When the cluster number is humanly selected as φ, the objective is to cluster the given data set into φ set such that similar points are grouped into the same cluster. In this article, the architecture of RBFNs is generated with the procedure of NFCM and its default value of the clustering number (φ) will determine the hidden-layer number of the RBFs. It is clear that the proposed configuration in RBFNs system is based on the condition $m = \varphi$ and the selected ith cluster center decide the respective c_i and w_i for the ith RBFs. Let $X = \{ x_j, j = 1, 2, ..., M\}$ denotes the training data set and each possible data point $x_j = (x_{j1}, x_{j2}, ..., x_{jv})$ in $X \subset R^v$ is the v-dimensional vector. If $U = \{\mu_{ij}, i = 1, 2, ..., m, j = 1, 2, ..., M\}$ is an initial fuzzy partition matrix, satisfied following conditions: $\sum_{i=1}^{m} \mu_{ij} = 1$, $j = 1, 2, ..., M$ and $0 < \sum_{j=1}^{M} \mu_{ij} < M$, $i = 1, 2, ..., m$. The definition of the objective function (J) is described by

$$J(U, z_1, z_2, ..., z_m) = \sum_{i=1}^{m}\sum_{j=1}^{M} (\mu_{ij})^{\lambda} \cdot \left(1 - \prod_{t=1}^{v} \exp\left(\frac{-(x_{j,t} - z_{i,t})^2}{2\sigma_t^2}\right)\right), \lambda > 1 \quad (3)$$

where, the coefficient parameter $\lambda > 1$ regulates the partition fuzziness degree and $Z = \{z_i, i = 1, 2, ..., m\}$ is the prototype of the cluster which represents the initial structure of the RBFNs. The purpose of the NFCM is to determine the proper Z which minimizes the objective function (J), and the necessary conditions for Eq. (3) to reach its minimum are

$$z_i = \frac{\sum_{j=1}^{M} (\mu_{ij})^{\lambda} \cdot \prod_{t=1}^{v} \exp\left(\frac{-(x_{j,t} - z_{i,t})^2}{2\sigma_t^2}\right) \cdot x_j}{\sum_{j=1}^{M} (\mu_{ij})^{\lambda} \cdot \prod_{t=1}^{v} \exp\left(\frac{-(x_{j,t} - z_{i,t})^2}{2\sigma_t^2}\right)}, \quad (4)$$

and

$$\mu_{ij} = \frac{\left[1 / \left(1 - \prod_{t=1}^{v} \exp\left(\frac{-(x_{j,t} - z_{i,t})^2}{2\sigma_t^2}\right)\right)\right]^{1/(\lambda-1)}}{\sum_{i=1}^{m} \left[1 / \left(1 - \prod_{t=1}^{v} \exp\left(\frac{-(x_{j,t} - z_{i,t})^2}{2\sigma_t^2}\right)\right)\right]^{1/(\lambda-1)}}, \quad j = 1, 2, ..., M \quad (5)$$

where $\sigma_t = \sqrt{\frac{\sum_{j=1}^{M} (x_{j,t} - \bar{x}_t)^2}{(N-1)}}$, $and\ \bar{x}_t = \frac{1}{M}\sum_{j=1}^{M} x_{j,t}$, $t = 1, 2, ..., v$. (6)

It notes that Eqs. (3)- (5) can be considered the new evaluated function to overcome the traditional FCM Euclidean norm type. The new distance function is described as follows:

$$d^2(x_j, z_i) = 1 - \prod_{t=1}^{v} \exp\left(\frac{-(x_{j,t} - z_{i,t})^2}{2\sigma_t^2}\right), \forall x, z \in R^v. \quad (7)$$

Two given training data set distributed around spherical and ellipsoidal region in 2-dimensional space are considered to explain the effect of two different distance function. For the spherical type data set, contours of the Euclidean norm type, i.e., $d(x_j, z_i) = \left\| x_j - z_i \right\|$ and the proposed new measured function are described in Figure 2(a) and Figure 2(b), respectively. From previous views of two measured profiles, it is easy to understand that Euclidean norm and the proposed measure functions can match the discussed data set to let them have the estimating ability for such spherical-like data set. In next case, the ellipsoidal-like data set is used to test separating abilities of traditional Euclidean norm and new functions. Their simulation results are respectively shown in Figure 2(c) and Figure 2(d). Obvious circumstances for their differences are that the Euclidean norm function gets a circle mapping but the new evaluated function makes an ellipsoidal-like one. The reasonable point from this experiment is that the proposed new distance function can adapt to measure the different type data set based on the exponent type measured function (Eq. (7)) and the calculation of the proposed factor σ_t (Eq. (6)).

The original data set shown in Figure 3(a) is intuitively partitioned into two unequal size clusters, which is applied to demonstrate robust analyses for FCM and NFCM. Figure 3(b) shows the clustering result of FCM whose clustering center is [(0.4744, 0.4399), (0.9356, 0.9738)]. Let us add one outlier point whose position is (6, 6) to the original data set, then the clustering center via FCM shown in Figure 3(d), becomes [(0.5781, 0.5594), (0.9870, 1.0190)]. Same experiment for NFCM is shown in Figure 3(c) and Figure 3(e), respectively. From Figure 3(c) and Figure 3(e), the detected clustering centers with NFCM are [(0.4564, 0.4219), (0.9203, 0.9540)] and [(0.4747, 0.4313), (0.9473, 0.9850)] with respect to no outlier and one outlier conditions. Simulation results show that the FCM learning algorithm is very sensitive to the outliers, but the suggested NFCM can adapt to the additional noise. In addition to previous example, we give some changes in original data set, whose value for one dimension is proportionally scaled with 5 and the other one is keeping in the same scale, to

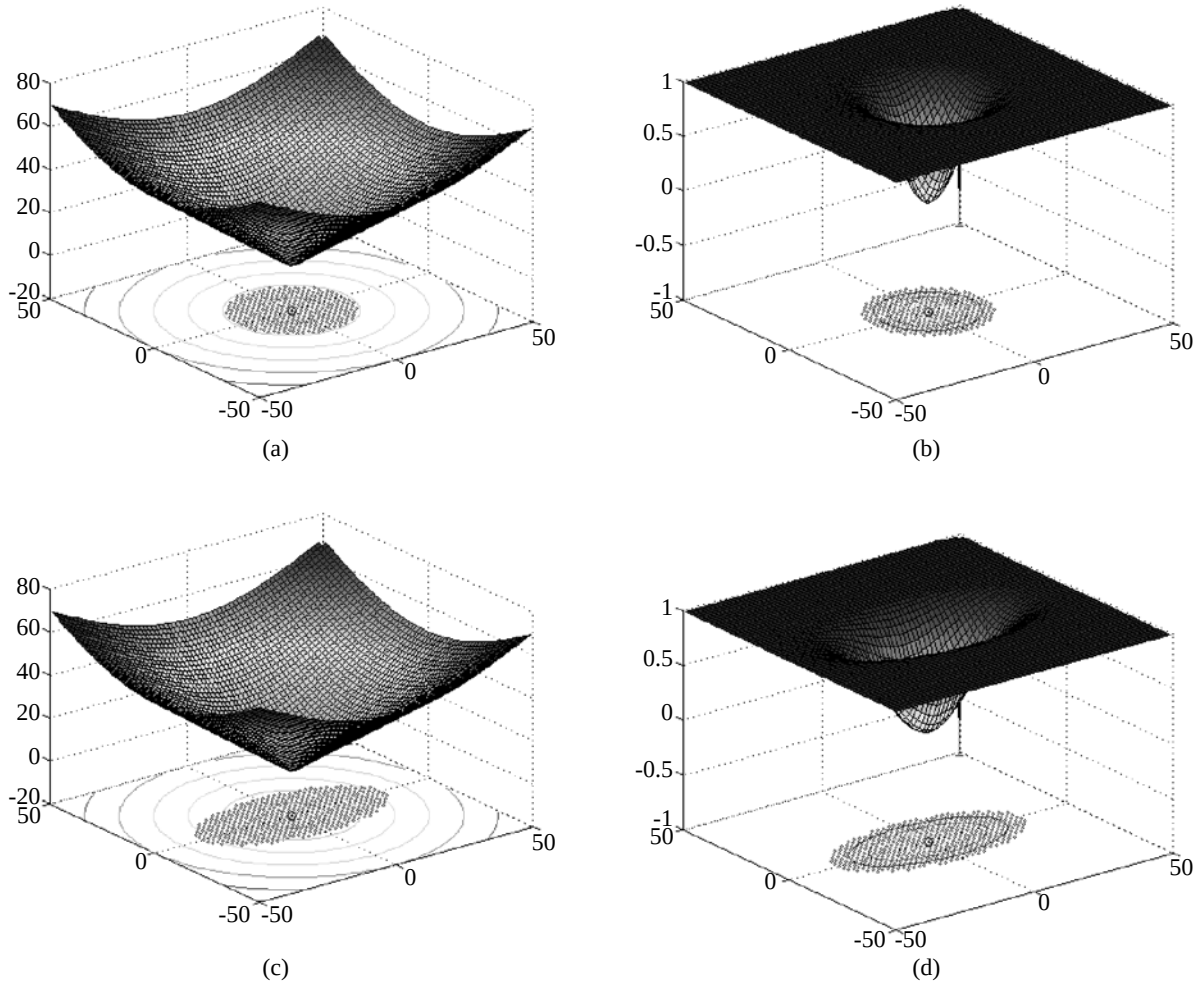


Fig. 2. Examples of distance functions, (a) Euclidean distance (for spherical cluster); (b) the proposed distance (for spherical cluster); (c) Euclidean distance (for ellipsoidal cluster); (d) the proposed distance (for ellipsoidal cluster).

Table 1. Clustering results for FCM and NFCM

	Center coordinates (no outlier)	Center coordinates (added outlier)	Center coordinates (1^{st} dimension is scaled with 5)
FCM	[(0.4744, 0.4399), (0.9356, 0.9738)]	[(0.5781, 0.5594), (0.9870, 1.0190)]	[(2.2818, 0.4219), (4.6014, 0.9540)]
NFCM	[(0.4564, 0.4219), (0.9203, 0.9540)]	[(0.4747, 0.4313), (0.9473, 0.9850)]	[(2.7829, 0.6659), (5.0910, 0.9518)]

demonstrate the efficiency of the proposed NFCM clustering method. Figure 3(f) and Figure 3(g) illustrate the clustering result of the FCM and NFCM, respectively. From Figure 3(f), it is clear that the final clustering center position is wrong. The other simulation with NFCM is shown in Figure 3(g), which demonstrates that the proposed NFCM can deter the mistake. Detailed information is described in Table 1. Based on the previous experiment, the NFCM has a powerful ability to reduce the effect of the scale and adapt the outlier. In a word, the NFCM shows a better adaptability than the popular FCM clustering method.

In this article, these derived clusters by the NFCM would be assigned to the center of the radial basis function c_i (i. e., $c_i = (z_{i1}, z_{i2}, ..., z_{iv-1})$) and the connection weights wi (i. e., $w_i = z_{iv}$) for the initial RBFNs. But

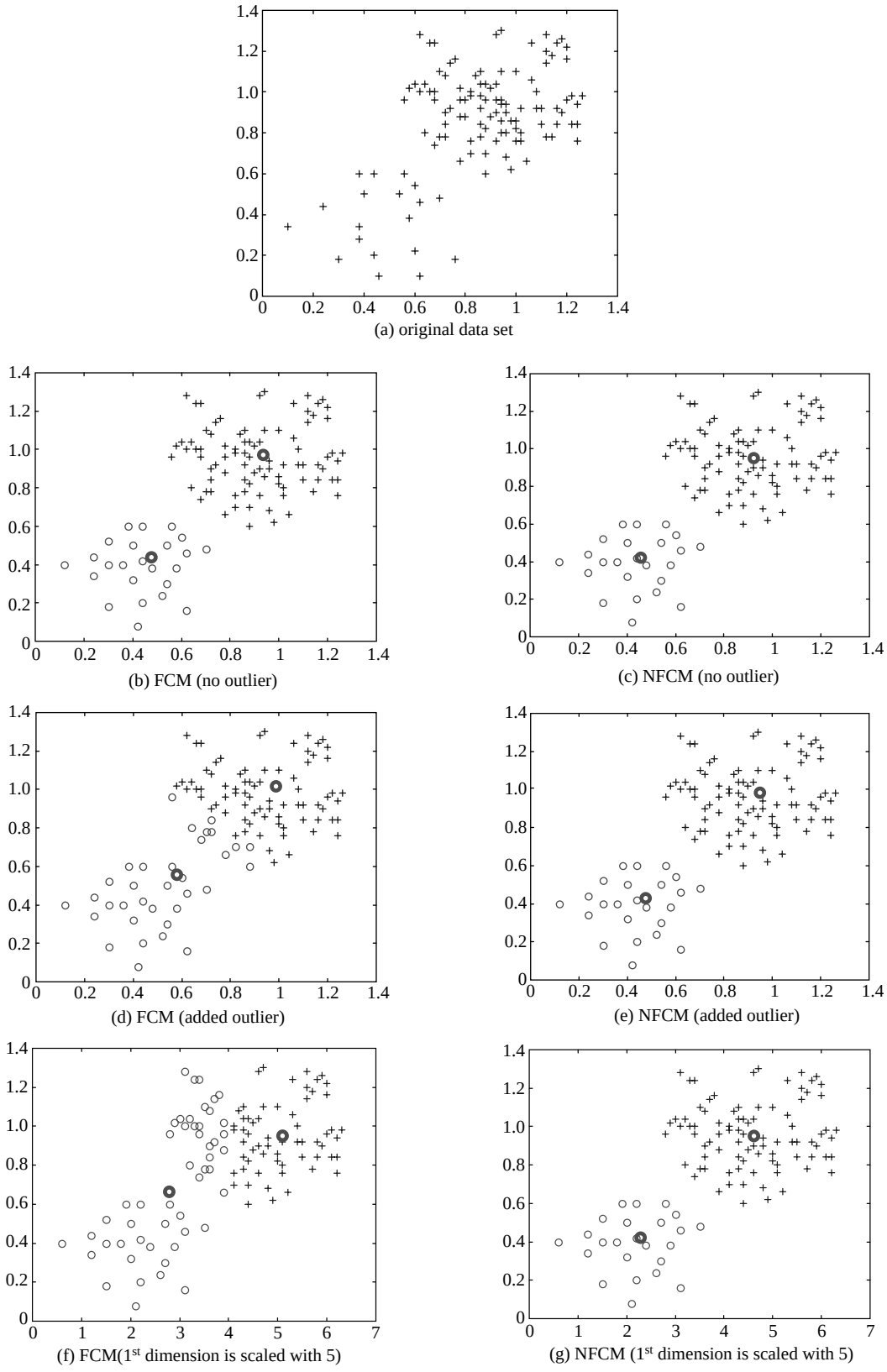


Fig. 3. Simulations comparison for FCM and NFCM methods.

δ_i are determined by a random operation. Every particle's position (i. e. solution) is made up of parameter set $\{c_{i1}, c_{i2}, ..., c_{in}, w_i, \delta_i, 1 \le i \le m\}$, but its velocity is initiated by a random process.

The initial structure of the RBFNMS is obtained by the NFCM algorithm, and the final solution will be refined by PSO and RLS learning algorithm. At the PSO learning cycle, each particle's position and velocity are updated by two best values. The first best one called *Lbest* is every particle's best solution that it has achieved so far. Another best one called *Gbest* is obtained by choosing the overall best value from all particles. It reveals that all particles share the best knowledge of optimal solutions. At each iteration step, the velocity of the particle is modified according to the relative data of Lbest and Gbest. The new velocity for each particle is updated by the following Eq.:

$$V_{p,d}(t+1) = \tau \times V_{p,d}(t) + \gamma_1\alpha_1 (Lbest_{p,d}(t) - \Phi_{p,d}(t)) + \gamma_2\alpha_2 (Gbest_d(t) - \Phi_{p,d}(t)) \quad (8)$$

where $V_{p,d}$ is the responding velocity of the pth particle in the dth dimension space and $\Phi_{p,d}$ is the responding solution of pth particle in the dth dimension space. Here, p is the index of particles; t represents current state, $t + 1$ represents the next time step; α_1 and α_2 are acceleration constant; γ_1 and γ_2 are random number between 0 and 1, and τ is the scaling factor to regulate the learning rate.

Since every particle's velocity is determined, the particle's position (i. e. solution) will be modified at the next time step by

$$\Phi_{p,d}(t+1) = \Phi_{p,d}(t) + V_{p,d}(t+1) \quad (9)$$

Based on Eqs. (8) and (9), the direction of every particle will update its original flying path and go gradually toward the direction of the best solution (*Gbest*), and it also learns the experience by their previous best solution (*Lbest*). In the previous description of the PSO learning way, it is shown that the computation of PSO is a slight load and the implementation is simple to execute.

The RLS learning algorithm has been performed an efficient way to approach an optimization [18, 20]. In the RLS learning cycle, it is acquired to modify the connection weights of the RBFNs. The main idea of this RLS learning algorithm is that the outputs of the constructed RBFNs are all approximate to those of the identified nonlinear functions (or call it desired outputs (y^d)). In this paper, this output of the RBFNs can be described from Eq. (2) as follows:

$$y = \sum_{i=1}^{m} q_i \cdot w_i \quad (10)$$

where q_i is the normalized activation of the ith RBFs corresponding to the input vector x and is defined by

$$q_i = \frac{HE_i(x)}{\sum_{i=1}^{m} HE_i(x)} \quad (11)$$

The Eq. (10) can be represented into matrix form, it is

$$y = Q\omega \quad (12)$$

where

$$\omega = [w_1, w_2, ..., w_m]^{\mathrm{T}} \subset R^{m \times 1} \quad (13)$$

$$Q = [q_1, q_2, ..., q_m] \subset R^{1 \times m} \quad (14)$$

In order to fastly adjust the connection weights of the RBFNs for approximating to the desired output (y^d), we use the RLS to determine the conncetion weights in the form of ω. The algorithm empowers to calculate the new $\omega(k+1)$ value on the base of training data pairs and the known parameter $\omega(k)$. Let initial time step be $k = 0$, and then $\omega(k+1)$ is modified by the following recursive iterations:

$$\Im(k+1) = \Im(k) - \frac{\Im(k) \cdot Q^T(k+1) \cdot Q(k+1) \cdot \Im(k)}{1 + Q(k+1) \cdot \Im(k) \cdot Q^T(k+1)}, \quad k = 0, 1, ..., M-1, \quad (15)$$

$$\omega(k+1) = \omega(k) + \Im(k+1) \cdot Q^T(k+1) \cdot (y^d(k+1) - Q(k+1) \cdot \omega(k)), \quad k = 0, 1, ..., M-1. \quad (16)$$

In this paper, the initial value $Q(0)$ = zero (zero represent the zero vector) and $\Im(0) = \eta$I, where η is a positive large number (= 100), M is the number of the training data and I is an $m \times m$ identity matrix. After M patterns calculation by Eqs. (15) and (16), those connection weights (ω) of the RBFNs are estimated recursively by the RLS algorithm.

The block diagram of the RBFNMS by the proposed HRPSO learning algorithm is plotted in Figure 4, which is summarized by following steps:

Step 1: Set the number of RBFs (m), and generate initial RBFs from the training data set to configure the initial structure of the RBFNs by the NFCM

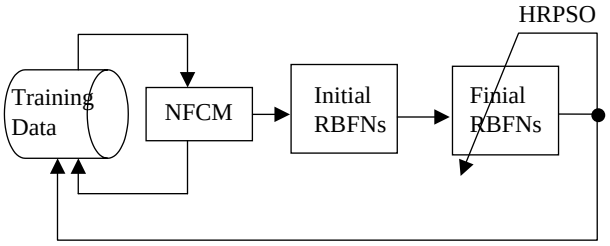


Fig. 4. Learning diagram of the RBFNMS.

clustering algorithm.

Step 2: Set the number of iterations (G), the value of the scaling factor (τ), the constant value (α_1, α_2), and the number of particles (P).

Step 3: Calculate the fitness value of each particle, and select the next personal best solution ($Lbest_p$) with the valuation of the fitness function. If the fitness value of particle Φ_p is better than the current fitness value of the personal best solution, then we set Φ_p to be the next personal best solution, otherwise the $Lbest_p$ remains the same.

Step 4: Compare each personal best value ($Lbest_p$) with the best global particle value (*Gbest*). If the fitness value of personal best solution is better than the current value of the global best solution (*Gbest*), then the evaluated R will be set to *Gbest*, otherwise the *Gbest* is not changed.

Step 5: Find the parameter set (R) by the PSO learning algorithm. For every particle, update its own velocity and position value according to Eqs. (8) and (9).

Step 6: Refine the conncetion weights (ω) by the proposed RLS learning algorithm to derive the final RBFNMS.

Step 7: If the termination conditions are satisfied, then go to exit, otherwise repeat step 3 to step 6.

Step 8: The best solution will be selected to build the desired RBFNMS.

ILLUSTRATED EXAMPLES

In order to verify the efficiency of the proposed HRPSO learning algorithm, two non-linear identification functions are presented in this section. The root-mean-square error (RMSE) of the training data is determined to measure the performance of the RBFNMS. The RMSE is calculated by

$$RMSE = \left(\frac{1}{M} \sum_{j=1}^{M} \left(y_j^d - \frac{\sum_{i=1}^{m} HE_i(x_j) \cdot w_i}{\sum_{i=1}^{m} HE_i(x_j)} \right)^2 \right)^{1/2} \quad (17)$$

Table 2. Parameter selection by HRPSO for Example 1

	c_{i1}	c_{i2}	δ_i	w_i
$i = 1$	-0.4981	0.5033	0.2518	-3.6631
$i = 2$	0.5047	0.5001	0.2556	4.0534
$i = 3$	2.9487	0.5432	14.0083	0.1553
$i = 4$	1.0325	0.4255	1.4030	-0.6170
$i = 5$	-2.8289	0.4992	5.6896	0.3045

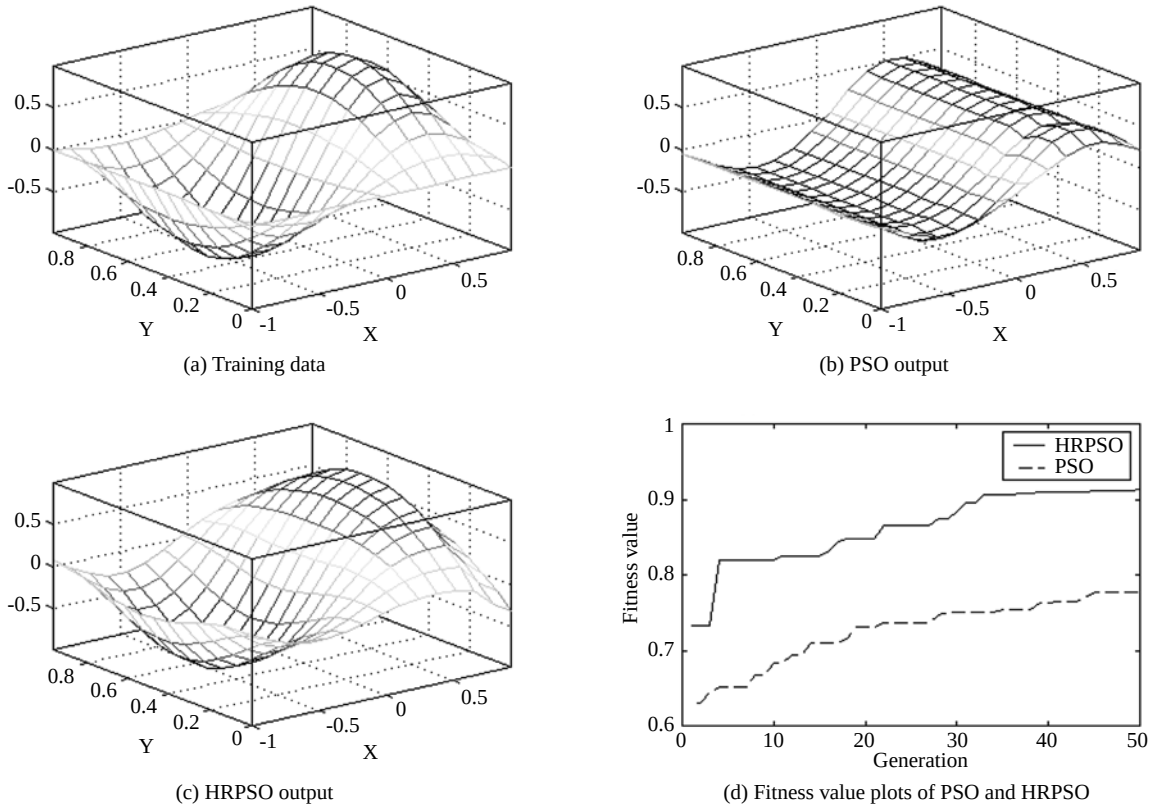
After above structure configuration is performed, the proposed HRPSO learning algorithm will be applied to tune the finial parameter set to minimize the *RMSE*. In this study, the fitness function is defined as exp (– *RMSE*), and so the goal of HRPSO is to maximize the fitness function vale (i. e., to minimize the *RMSE*).

Example 1: Modeling a Function

In this case, the RBFNMS is determined to approximate the nonlinear function, which is defined by [13, 20]

$$F_1 = \sin(\pi x_1) \cdot \sin(\pi x_2) \quad (18)$$

225 pieces of training data are uniformly distributed in the range of $x_1 \in [-1, 1]$ and $x_2 \in [0, 1]$. The required parameter setting is as follows: the partition fuzziness degree parameter $\lambda = 2$, the scaling factor $\tau = 0.75$, the $\alpha_1 = 1.5$ and $\alpha_2 = 1.5$, the number of particles (P) is 5, the number of iterations (G) is 50, and the number of clusters (i. e. the number of RBFs) $m = 5$. In the first NFCM step, 5 cluster centers are selected as the initial structure of the RBFNs. Following the proposed PSO and RLS learning procedure, which is described in Section 3, the final parameters are shown in Table 2. Figure 5 demonstrates computer simulation results, where (a), (b), and (c) show the output of training data, the output of the RBFNMS by only the PSO method, and the output of RBFNMS by the proposed HRPSO algorithm, respectively; (d) shows the best (highest) fitness value against the iteration number for HRPSO (solid curve) and PSO (dash curve). Simulation results show that the fitness value of the HRPSO quickly raises to the higher fitness value (i. e., HRPSO can fastly approach the desired output). The performance comparison for different modeling learning methods with MSE (mean-square error) is listed in Table 3. From Table 3, it is shown that best MSE by HRPSO is obviously smaller than that of PSO. The proposed HRPSO method can approximate the unknown function with much better accuracy than other previous study [13, 20]. It is clear that the constructed HRPSO learning algo-



(a) Training data (b) PSO output

(c) HRPSO output (d) Fitness value plots of PSO and HRPSO

Fig. 5. $\sin(\pi x_1) \cdot \sin(\pi x_2)$ function approximation, (a) training data output; (b) output of RBFNMS by PSO; (c) output of RBFNMS by HRPSO; (d) fitness value against iteration by PSO and HRPSO.

Table 3. Performance comparisons with different methods.

Model	No. of RBFs/ No. of rules	*MSE*	*RMSE*
PSO	5	0.1786	0.4226
HRPSO	5	0.0020	0.0447
Wong's System [12]	6	0.0042	0.0648
Lee's System [13]	6	0.0027	0.0520

Note: The last two rows are from [14].

rithm with small RBFNs and small swarm size is enough to efficiently generate the desired RBFNMS.

Example 2: Modeling a discrete dynamic system

In order to illustrate the efficiency of the HRPSO, the objective of the RFFNMS is to approximate the nonlinear function [17] described as

$$F_2 = (1 + x_1^{-2} + x_2^{-1.5})^2, \; x_1, x_2 \in [1, 5] \qquad (19)$$

In this example, training data is uniformly distributed in the range of $x_1 \in [1, 5]$ and $x_2 \in [1, 5]$. To train the RBFNMS by the proposed HRPSO learning algorithm, x_1 and x_2 are uniformly distributed in the range of $x_1 \in [1, 5]$ and $x_2 \in [1, 5]$ and then use Eq. (19) to generate 400 training data pairs. The required parameter setting is as follows: the partition fuzziness degree parameter $\lambda = 2$, the scaling factor $\tau = 0.75$, the $\alpha_1 = 1.5$ and $\alpha_2 = 1.5$, the number of particles (P) is 5, the number of iterations (G) is 100, and the number of clusters (i. e. the number of RBFs) $m = 5$. After 100 training epochs, The RBFNs with 5 RBFs are enough to modeling the nonlinear function by the HRPSO learning algorithm. The simulation result for training data, PSO, and HRPSO are shown in Figure 6(a), Figure 6(b), and Figure 6(c), respectively. The best fitness value against the iteration for HRPSO (solid curve) and PSO (dash curve) are shown in Figure 6(d). The simulation results show that the HRPSO-based RBFNMS can efficiently rebuild the nonlinear function. Final parameter values which are derived by HRPSO are shown in Table 4. Performance comparisons for PSO, HRPSO, and Sugeno's System are listed in Table 5. From Table 5, it is clear that the result obtained by HRPSO has greater improvement than those of standard PSO and Sugeno's method.

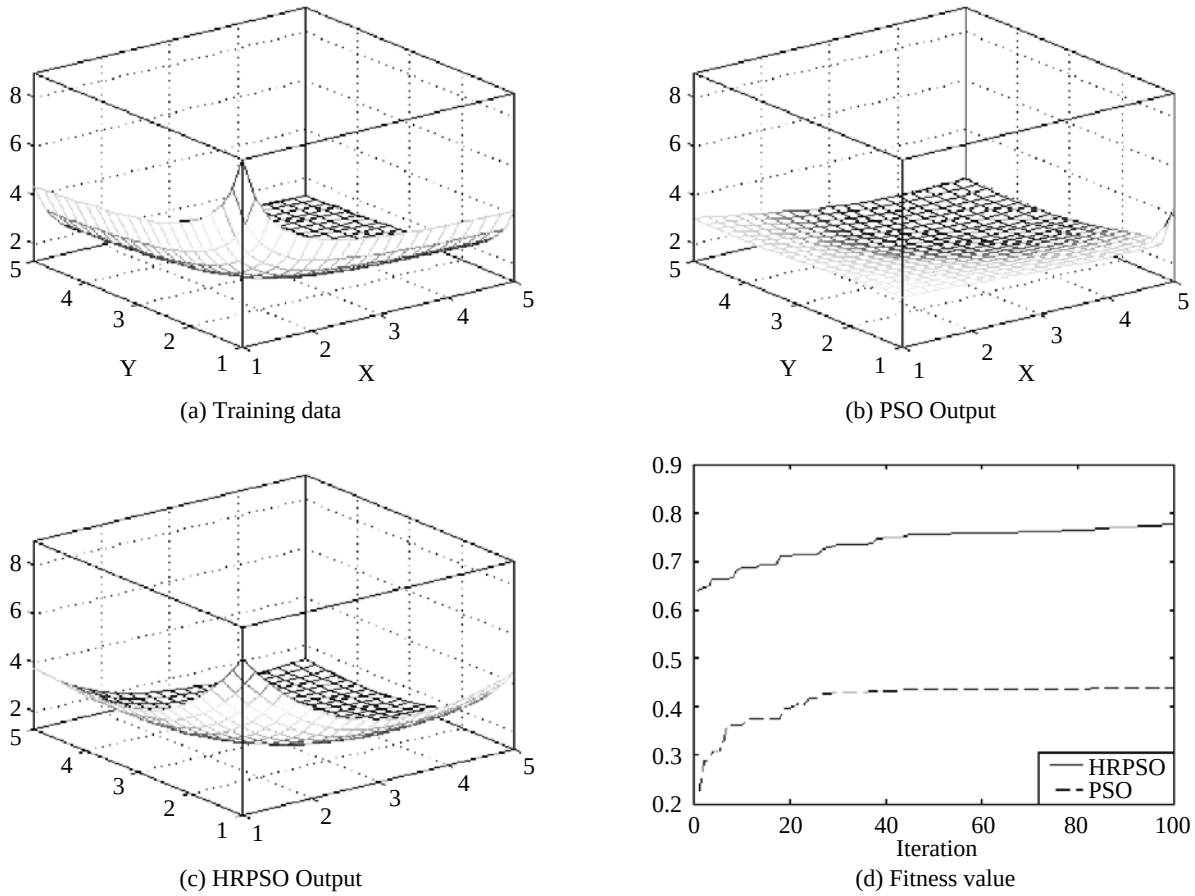


Fig. 6. Discrete Dynamic System approximation, (a) training data; (b) Output by PSO; (c) Output by HRPSO; (d) fitness value against iteration in PSO and HRPSO method.

Table 4. Parameter values by HRPSO for Example 2

	c_{i1}	c_{i2}	δ_i	w_i
$i = 1$	3.5759	3.1393	1.2722	8.1104
$i = 2$	2.7588	2.4217	1.7085	-9.5032
$i = 3$	1.3123	3.0799	5.3586	25.0336
$i = 4$	5.1000	5.1000	1.5843	-5.0275
$i = 5$	2.8146	5.1000	3.0947	-16.9577

Table 5. Performance comparisons with different methods for Example 2

Model	No. of RBFs/ No. of rules	*MSE*	*RMSE*
PSO	5	0.6798	0.8245
HRPSO	5	0.0648	0.2546
Sugeno's System [17]	6	0.079	0.2811

CONCLUSIONS

In this study, we use a set of Gaussian function to define RBFs. A small number of RBFs can be clustered by the NFCM to develop initial structure of RBFNMS, which generate a preliminary definition of RBFNs. Furthermore, the proposed HRPSO learning algorithm is performed to fastly train the desired RBFNMS. The HRPSO can simultaneously tune the parameter set $\{c_{i1}, c_{i2}, ..., c_{in}, w_i, \delta_i, 1 \le i \le m\}$, of RBFNs for the design of RBFNMS. Two nonlinear approximation problems are applied to illustrate the efficiency of the proposed HRPSO learning algorithm. In those illustrated examples, demonstrations show that only a fewer particles with small number of RBFs are enough to perform the better task for identifying nonlinear modeling problems than other previous study.

REFERENCES

1. Akhmetov, D.F., Dote, Y., Ovaska, S.J., "Fuzzy Neural Network with General Parameter Adaptation for Modeling of Nonlinear Time-Series," *IEEE Transactions on Neural Networks*, Vol. 12, No. 1, pp. 148-152 (2001).

2. Behloul, F., Lelieveldt, B.P.F., Boudraa, A., and Reiber, J.H.C., "Optimal Design of Radial Basis Function Neural Networks for Fuzzy-Rule Extraction in High Dimensional Data," *Pattern Recognition*, Vol. 35, No. 3, pp. 659-675 (2002).
3. Bezdek, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York (1981).
4. Choi, S.W.D., Lee, J.H., Park, I., Lee, B., "Nonlinear Regression Using RBFN with Linear Submodels," *Chemometrics and Intelligent Laboratory Systems*, Vol. 65, No. 2, pp. 191-208 (2003).
5. Chuang, C.C., Jeng, J.T., and Lin, P.T., "Annealing Robust Radial Basis Function Networks for Function Approximation with Outliers," *Neurocomputing*, Vol. 56, pp. 123-139 (2004).
6. Ciftcioghu, O., "GA with Orthogonal Transformation for RBFN Configuration," *Proceeding of IEEE International Conference on Neural Networks*, pp. 1934-1939 (2002).
7. Clerc, M. and Kennedy, J., "The Particle Swarm-Explosion, Stability, and Convergence in A Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, pp. 58-73 (2002).
8. Dunn, J.C., "A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact, Well-Separated Clusters," *Journal of Cybernetics*, Vol. 3, pp. 32-57 (1974).
9. Gaing, Z.L., "A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System," *IEEE Transactions on Energy Conversion*, Vol. 19, No. 2, pp. 384-391 (2004).
10. Juang, C.F., "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 34, No. 2, pp. 997-1006 (2003).
11. Kennedy, J., "The Particle Swarm: Social Adaptation of Knowledge," *Proceeding of International Conference on Evolutionary Computation*, Indianapolis, IN, pp. 303-308 (1997).
12. Kennedy, J. and Eberhart, R.C., "Particle Swarm Optimization," *Proceeding of IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1942-1948 (1995).
13. Lee, S.J. and Ouyang, C.H., "A Neuro-Fuzzy System Modeling with Self-Constructing Rule Generation and Hybrid SVD-Based Learning," *IEEE Transactions on Fuzzy Systems*, Vol. 11, No. 3, pp. 341-353 (2004).
14. Naka, S., Genji, T., Yura, T., and Fukuyama, Y., "A Hybrid Particle Swarm Optimization for Distribution State Estimation," *IEEE Transactions on Power Systems*, Vol. 18, No. 1, pp. 60-68 (2003).
15. Nam, M.D. and Thanh, T.C., "Approximation of Function and its Derivatives Using Radial Basis Function Networks," *Applied Mathematical Modeling*, Vol. 27, No. 3, pp. 197-220 (2003).
16. Park, J. and Sandberg, I.W., "Approximation and Radial Basis Function Networks," *Neural Computation*, Vol. 5, pp. 305-316 (1993).
17. Sugeno, M. and Yasukawa, T., "Fuzzy-Logic-Based Approach to Qualitative Modeling," *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 1, pp. 7-31 (1993).
18. Wang, L.X., *A Course in Fuzzy Systems and Control*, Prentice Hall, Inc., Englewood Cliffs, NJ (1997).
19. Wang, W.Y. and Li, W.H., "Evolutionary Learning of BMF Fuzzy-Neural Networks Using a Reduced-form Genetic Algorithm," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 33, No. 6, pp. 966-976 (2003).
20. Wong, C.C. and Chen, C.C., "A Hybrid Clustering and Gradient Descent Approach for Fuzzy Modeling," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 29 pp. 686-693 (1999).
21. Wong, C.C. and Chen, C.C., "A GA-Based Method for Constructing Fuzzy Systems Directly from Numerical Data," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 30, No. 6, pp. 905-911 (2000).