An Efficient Symbiotic Particle Swarm Optimization for Recurrent Functional Neural Fuzzy Network Design

Cheng-Jian Lin and Chi-Feng Wu

Abstract

In this paper, a recurrent functional neural fuzzy network (RFNFn) with symbiotic particle swarm optimization (SPSO) is proposed for solving identification and prediction problems. The proposed RFNFn model has feedback connections added in the membership function layer that can solve temporal problems. Moreover, an efficient learning algorithm, called symbiotic particle swarm optimization (SPSO), combined symbiotic evolution and modified particle swarm optimization for tuning parameters of the RFNFn. Simulation results show that the converging speed and root mean square error (RMS) of the proposed method has a better performance than those of other methods.

Keywords: Neural fuzzy networks, recurrent networks, particle swarm optimization, symbiotic evolution, identification, prediction.

1. Introduction

Neural fuzzy networks [1]-[4] have become a popular research topic. Two typical neural fuzzy networks are the Mamdani-type and the TSK-type. Many researchers [2]-[3] have shown that TSK-type neural fuzzy networks offer better network size and learning accuracy than Mamdani-type neural fuzzy network. In the typical TSK-type neural fuzzy network, which is a linear polynomial of the input variables, the model output is approximated locally by the rule hyperplanes. Nevertheless, the traditional TSK-type neural fuzzy network does not take full advantage of the mapping capabilities that may be offered by the consequent part. Introducing a nonlinear function, especially a neural structure, to the consequent part of the fuzzy rules has yielded the NARA [5] and the CANFIS [6] models. These models [5]-[6] apply multilayer neural networks into the consequent part of the fuzzy rules. Although the interpretability of the model is reduced, the representational capability of the model is markedly improved. However, the multilayer neural network has such disadvantages as slower convergence and greater computational complexity. In [7], we use the functional link neural network (FLNN) in the consequent part of the fuzzy rules. The FLNN [8]-[9] is a single layer neural structure capable of forming arbitrarily complex decision regions by generating nonlinear decision boundaries. Moreover, using the functional expansion effectively increases the dimensionality of the input vector and the hyperplanes generated by the FLNN provides better discrimination capability in input data space.

For a dynamic system, the output is a function of past inputs or past outputs or both, identification of this kind of system is not as direct as a static system, and to deal with temporal problems of a dynamic system, the recurrent neural network and the recurrent neuro-fuzzy system have attracted great interest. Hence, for nonlinear system processing, the most commonly used model is the neural network or the neuro-fuzzy system. If a feed-forward network is adopted for this task, then we should know the number of delayed inputs and outputs in advance and feed these delayed inputs and outputs as a tapped line into the network input. The problem of this approach is that the exact order of the dynamic system is usually unknown. To solve this problem, interest in using recurrent networks for processing dynamic system has been stably growing in recent years[10]-[11]. In this paper, a recurrent functional neural fuzzy networks (RFNFn) is proposed. The RFNFn is a recurrent multiplayer connectionist network for fuzzy reasoning and can be constructed from a set of fuzzy rules. In the RFNFn, adding feedback connections in the second layer develops the temporal relations. So it can deal with temporal problems.

Training of the parameters is the main problem in designing a neural fuzzy system. Genetic algorithms (GAs), powerful tools based on biological mechanisms and natural selection theory [12], have received considerable attention regarding its potential as an optimization technique for complex problems and have been successfully applied in various areas [13]. A genetic algorithm (GA)
is a parallel, global search technique that emulates operators. Because it simultaneously evaluates many points in the search space, it is more likely to converge toward the global solution. But GAs have two main drawbacks. One is lack of the local search ability and the other is the premature convergence [14]. Therefore, in recent years, some researchers [13]-[15] have proposed various improved-GAs to solve global optimization problems. In 1995, a new optimization algorithm, called particle swarm optimization (PSO) was developed by Kennedy and Eberhart [16]. The underlying motivation for the development of PSO algorithm is the social behavior of animals, such as bird flocking, fish schooling and swarm theory. The PSO has come to be widely used as a problem solving method in engineering and computer science. It is not only a recently invented high-performance optimizer that is very easy to understand and implement, but also requires less computational bookkeeping. But the PSO have one main drawback that convergence speed is too slow. Therefore, in this study, we proposed a symbiotic particle swarm optimization algorithm (SPSO) for solving the above-mentioned problems.

The advantages of the proposed RFNFN model with SPSO learning method are summarized as follows. The RFNFN model can deal with temporal problems effectively.

(1) The SPSO learning method adopts a subgroup symbiotic evolution strategy which uses the rule-based subgroup to evolve separately.

(2) It performs better and converges more quickly than some traditional genetic methods.

This paper is organized as follows: Section 2 describes the structure of the recurrent functional neural fuzzy network. An efficient symbiotic particle swarm optimization learning algorithm is proposed in Section 3. Section 4 presents the results of the simulation with optimization learning algorithm is proposed in Section 3. The rest of this section details these structures.

2.1 Functional Link Neural Networks

The functional link neural network is a single layer network in which the need for hidden layers is eliminated. While the input variables generated by the linear links of neural networks are linearly weighted, the functional link acts on an element of input variables by generating a set of linearly independent functions, which are suitable orthogonal polynomials for a functional expansion, and then evaluating these functions with the variables as the arguments. Therefore, the FLNN structure considers trigonometric functions. For example, for a two-dimensional input \( X = [x_1, x_2]^T \), enhanced data are obtained using trigonometric functions as \( \phi = [1, x_1, \sin(x_1), \cos(x_1), x_2, \sin(x_2), \cos(x_2)]^T \). Thus the input variables can be separated in the enhanced space [17]. In the FLNN structure with reference to Fig. 1, a set of basis functions \( \phi \) and a fixed number of weight parameters \( W \) represent \( f_m(x) \). The theory behind the FLNN for multidimensional function approximation has been discussed elsewhere [18] and is analyzed below.

![Figure 1. Structure of FLNN.](image)

Consider a set of basis functions \( \mathbf{B} = \{\phi_i \in \Phi(A)\}_{i=k} \), \( \mathbf{K} = \{1, 2, \ldots\} \) with the following properties; 1) \( \phi_1 = 1, 2) \) the subset \( \mathbf{B}_j = \{\phi_i \in \mathbf{B}\}^M_{i=1} \) is a linearly independent set, meaning that if \( \sum_{i=1}^{M} w_i \phi_i = 0 \), then \( w_i = 0 \) for all \( k = 1, 2, \ldots, M \) and 3) \( \sup \left[ \sum_{i=1}^{M} \|\phi_i\|_2^2 \right] < \infty \).

Let \( \mathbf{B} = \{\phi_i\}_{i=1}^M \) be a set of basis functions to be considered, as shown in Fig. 1. The FLNN comprises \( M \) basis functions \( \{\phi_1, \phi_2, \ldots, \phi_M\} \in \mathbf{B} \). The linear sum of the \( j \)th node is given by

\[
\hat{y}_j = \sum_{k=1}^{M} w_{kj} \phi_k(X)
\]

where \( X \in \mathbb{A} \subset \mathbb{R}^N \), \( X = [x_1, x_2, \ldots, x_N]^T \) is the input.
vector and \( \mathbf{W}_j = [w_{j1}, w_{j2}, \ldots, w_{jM}]^T \) is the weight vector associated with the \( j \)th output of the FLNN. \( \hat{y}_j \) denotes the local output of the FLNN structure and the consequent part of the \( j \)th fuzzy rule in the RFNFN model. Thus, Eq. (1) can be expressed in matrix form as

\[
\hat{y}_j = \mathbf{W}_j \Phi_j,
\]

where \( \Phi_j = [\phi_1(x), \phi_2(x), \ldots, \phi_M(x)]^T \) is the basis function vector, which is the output of the functional expansion block. The \( m \)-dimensional linear output may be given by \( \hat{y} = \mathbf{W}\Phi \), where \( \hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m]^T \), \( m \) denotes the number of functional link bases, which equals the number of fuzzy rules in the RFNFN model, and \( \mathbf{W} \) is a \( (m \times M) \)-dimensional weight matrix of the FLNN given by

\[
\mathbf{W}_j = \begin{bmatrix} w_{j1} & w_{j2} & \cdots & w_{jM} \end{bmatrix}.
\]

2.2 Structure of the RFNFN Model

This subsection describes the RFNFN model, which uses a nonlinear combination of input variables as the consequent part of the fuzzy rules. Each fuzzy rule corresponds to a sub-FLNN, comprising a functional link. Figure 2 presents the structure of the proposed RFNFN model. The RFNFN model realizes a fuzzy if-then rule in the following form.

\[
\text{Rule}_j: \quad \text{IF } h_{ij} \text{ is } A_{ij} \text{ and } h_{ij} \text{ is } A_{ij} \ldots \text{ and } h_{ij} \text{ is } A_{ij} \text{ THEN } \hat{y}_j = \sum_{k=1}^{M} w_{ij} \phi_k
\]

\[
= w_{ij1} \phi_1 + w_{ij2} \phi_2 + \cdots + w_{ijM} \phi_M
\]

where \( h_{ij} \) and \( \hat{y}_j \) are the input pattern \( \{x_1, x_2, \ldots, x_n\} \) plus the temporal term for the linguistic term of the precondition part with Gaussian membership function \( A_{ij} = \{A_{ij1}, \ldots, A_{ijn}\} \) and local output variables, respectively; \( N \) is the number of input variables; \( w_{ij} \) is the link weight of the local output; \( \phi_k \) is the basis trigonometric function of input variables; \( M \) is the number of basis function, and Rule\(_j\) is the \( j \)th fuzzy rule.

In order to give a clear understanding of the mathematical function of each node, we will describe functions of RFNFN layer by layer. For notation convenience, \( u^{(i)} \) denotes the output of a node in the \( i \)th layer.

Layer 1: No function is performed in this layer. The node only transmits input values to layer 2.

\[
u^{(1)}_i = x_j.
\]

Layer 2: Nodes in this layer correspond to a single linguistic label of the input variables in layer 1. Therefore, the calculated membership value specifies the degree to which an input value

\[
u^{(2)}_j(t) = \exp \left\{ -\frac{\left|h_{ij} - m_{ij}\right|^2}{\sigma_{ij}^2} \right\}
\]

where \( m_{ij} \) and \( \sigma_{ij} \) are the mean and variance of the Gaussian membership function, respectively. Additionally, the input of this layer for the discrete time scan be denoted by

\[
h_{ij}(t) = u^{(1)}_i(t) + u^{(2)}_j(t-1) \cdot \theta_{ij}
\]

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

Layer 3: Nodes in this layer are called rule nodes and each represents one fuzzy logic rule. The links in layer 3 are used to perform precondition matching of fuzzy logic rules. Hence the rule nodes perform the fuzzy product operation,

\[
u^{(3)}_j = \prod_{i=1}^{N} u^{(2)}_j
\]

where the \( \prod_{i=1}^{N} u^{(2)}_j \) of a rule node represents the firing strength of its corresponding rule.

Layer 4: Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output from layer 3, and the other inputs are nonlinear combinations of input variables from a functional link neural network. For such a node,

\[
u^{(4)}_j = \prod_{k=1}^{M} w_{ij} \phi_k
\]
where \( w_{ij} \) is the corresponding link weight of functional link neural network and \( \phi_k \) is the functional expansion of input variables. The functional expansion used a trigonometric polynomial basis function, given by \( \{ x_i, \sin(\pi x_i), \cos(\pi x_i), x_i, \sin(\pi x_j), \cos(\pi x_j) \} \) for two-dimensional input variables. Therefore, \( M \) is the number of basis functions, \( M = 3 \times N \), where \( N \) is the number of input variables.

Layer 5: Each node in this layer corresponds to a single output variable. The node integrates all of the actions recommended by layer 3 and 4 and acts as a defuzzifier with,

\[
y = u^{(5)} = \sum_{j=1}^{R} \frac{\sum_{j=1}^{R} u^{(4)}_j (\sum_{k=1}^{M} w_{kj} \phi_k)}{\sum_{j=1}^{R} u^{(3)}_j} = \sum_{j=1}^{R} u^{(3)}_j \hat{y}_j
\]

where \( R \) is the number of fuzzy rules, and \( y \) is the output of the RFNFN model.

### 3. Learning Algorithms for the RFNFN Model

In this section, we will review the basic concepts of the particle swarm optimization and process of the symbiotic particle swarm optimization (SPSO), individually.

#### 3.1 Review Particle Swarm Optimization

PSO is a recently invented high performance optimizer that possesses several highly desirable attributes, including the fact that the basic algorithm is very easy to understand and implement. It requires less computational memory and fewer lines of code. Each particle has a velocity vector \( \vec{v} \) and a position vector \( \vec{x} \) to represent a possible solution. Each particle has three choices in evolution: (1) Insist on oneself. (2) Move towards the optimum itself at present. Each particle remembers its own personal best position that it has ever found, called the local best. (3) Move towards the best the population has met. Each particle also knows the best position found by any particle in the swarm, called the global best. The PSO reaches a balance among these three choices. Using the local best position (\( L_{best} \)) and the global best position (\( G_{best} \)), a new velocity for each particle is updated by

\[
\vec{v}_i(k+1) = \omega \cdot \vec{v}_i(k) + \phi_1 \cdot \text{rand}() \cdot (L_{best} - \vec{x}_i(k)) + \phi_2 \cdot \text{rand}() \cdot (G_{best} - \vec{x}_i(k))
\]

where \( \omega, \phi_1, \phi_2 \) are called the coefficient of inertia, cognitive and society study respectively. The \( \text{rand}() \) is uniformly distributed random numbers in \([0, 1]\). The term \( \vec{v}_i \) is limited to the range \( \pm \vec{v}_{\text{max}} \). If the velocity violates this limit, it will be set at its proper limit. The concept of the updated velocity is illustrated in Fig. 3.

Changing velocity enables every particle to search around its individual best position and global best position. Based on the updated velocities, each particle changes its position according to the following:

\[
\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1)
\]

When every particle is updated, the fitness value of each particle is calculated again. If the fitness value of the new particle is higher than those of local best, then the local best will be replaced with the new particle.

#### 3.2 The proposed symbiotic particle swarm optimization Learning Algorithm

This subsection describes the proposed evolutionary learning algorithm which is combine the symbiotic evolution and modified particle swarm optimization, called symbiotic particle swarm optimization (SPSO) algorithm. The notion of symbiotic evolution [19]-[20] is similar to the implicit fitness sharing used in an immune system model. The authors evolve artificial antibodies to match or detect artificial antigens. Each antibody can only match a single antigen, and different antibodies are needed to protect effectively against various antigens. These antibodies consist in separating the population into subpopulations and changing the way fitness values are assigned by fitness sharing. An antibody is selected for replacement by randomly choosing a subset of the population and then selecting the member of that subset that is most similar to the new antibody. Therefore, summing the fitness values of all possible combinations of that antibody with other current antibodies and dividing the sum by the total number of combinations yields the fitness of an antibody.

The basic idea of symbiotic evolution is that an indi-
Individual is used to represent a single fuzzy rule. A fuzzy system is formed when several individuals, which are randomly selected from a population, are combined. With the fitness assignment performed by symbiotic evolution, and with the local property of a fuzzy rule, symbiotic evolution and the fuzzy system design can complement each other. The structure of the proposed particles in the symbiotic evolution is shown in Fig. 4.

The learning process of the proposed SPSO includes the coding, initialization, fitness assignment and sub-particles update. The coding step is concerned with the membership functions and fuzzy rules of a fuzzy system that represent particle for SPSO. The initialization step assigns the sub-particles values before the learning process begins. Then, the fitness value assignment gives a suitable value to each fuzzy rule during the learning process, the value higher the particles survive higher. Repeat until a given termination condition is met.

The whole learning process is described step by step below.

A. Production initial swarm

The coding step is related with the membership functions and fuzzy rules of a fuzzy system that represent sub-particles suitable for symbiotic evolution. Figure 6 shows an example of the coding of parameters of a fuzzy rule into a sub-particle, where \( i \) and \( j \) represents \( i \)th input variable and \( j \)th rule.

![Figure 4. Structure of the particles in the symbiotic evolution.](image)

![Figure 5. Flowchart of the proposed SPSO designs method.](image)

![Figure 6. Coding a fuzzy rule into a sub-particle in the proposed SPSO.](image)

The initialization step must assign value before the evolution process begins. The initial value assigns by the randomly in the range \([-1, 1]\) in each subgroup.

B. Fitness value assign step

The fitness value of a rule (a sub-particle) is computed by summing up the fitness values of all the feasible combinations of that rule with all other randomly selected rules and then dividing the sum by the total number of combinations. The details for assigning the fitness value are described step by step as follows.

Step 1: Randomly select \( R \) fuzzy rules (sub-particle) from each subgroup for composing fuzzy system.

Step 2: Evaluate every fuzzy system that is generated from step1 to obtain fitness value. The fitness value is designed according the following formula:

\[
m_i, \sigma_i, \theta_i, m_j, \sigma_j, \theta_j, \ldots, m_k, \sigma_k, \theta_k, w_1, w_2, \ldots, w_{ik}
\]
\[ F = \frac{1}{1 + \sqrt{\frac{1}{D} \sum_{d=1}^{D} (y_d - \overline{y}_d)^2}} \]  

(11)

Where \( y_d \) represents the \( d \)th model output; \( \overline{y}_d \) represents the \( d \)th desired output, and \( D \) represents the number of input data.

Step 3: Divide the fitness value by \( R \) and accumulate the divided fitness value to the selected \( R \) rules with their fitness value records that were set to zero initially.

Step 4: Repeat the above steps until each rule (sub-particle) in each subgroup has been selected a sufficient number of times, and record the number of fuzzy systems in which each sub-particle has participated.

Step 5: Divide the accumulated fitness value of each sub-particle by the number of times it has been selected.

C. Fitness value assign step

In this paper, we proposed the new velocity update function to move up the performance, the velocity of the sub-antibody is redefined as the following equation:

\[
v_{i,j}(k+1) = \omega \cdot v_{i,j}(k) + \phi_1 \cdot \text{rand}(k) \cdot (C_{best} - x_{i,j}(k)) + \phi_2 \cdot \text{rand}(k) \cdot (G_{best} - \overline{x}_{i,j}(k)) + \phi_3 \cdot \text{rand}(k) \cdot (L_{best} - \overline{x}_{i,j}(k))
\]

(12)

where \( \omega \), \( \phi_1 \), \( \phi_2 \) and \( \phi_3 \) are called the coefficient of inertia, cognitive, group and society study respectively. The \( C_{best} \) is changed according to rule. We hope that accelerate every sub-particle in a direction between the best self, the best of partial solution and the best of full solution so far.

4. Experiment Results

In this section, we discuss two different examples to evaluate the proposed RFNFN model with SPSO learning method. The first example is to identify a nonlinear dynamic system and the second example involves predicting a chaotic time series [21]. In example 1, the parameter learning through SPSO method of the coefficient \( w \) was set to 0.5, the cognitive coefficient \( \phi_1 \) and the group coefficient \( \phi_2 \) were set to 1, and the society coefficient \( \phi_3 \) was set to 1.5. The swarm sizes were set to 50. In examples 2, the parameter learning through SPSO method of the coefficient \( w \) was set to 0.3, the cognitive coefficient \( \phi_1 \) and the group coefficient \( \phi_2 \) were set to 1, and the society coefficient \( \phi_3 \) was set to 1.5. The swarm sizes were also set to 50. The above-mentioned parameters \( w \), \( \phi_1 \), \( \phi_2 \) and \( \phi_3 \), we are using the try-and-error methods to determine these parameter values.

Example 1: Identification of a Nonlinear Dynamic System

The systems to be identified are dynamic systems whose outputs are functions of past inputs and past outputs as well. For this dynamic system identification, since the RFNFN network is used, only the current state of system and the control signal are fed as input to the network. The adopted identification configuration is a serial-parallel model shown in Fig. 7.

![Figure 7. Series-parallel identification model with the RFNFN model.](image)

The plant to be identified in this example is guided by the following difference equation:

\[
y_{p}(k+1) = f(y_{p}(k), y_{p}(k-1), y_{p}(k-2), u(k), u(k-1))
\]

(13)

where

\[
f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1x_2x_3(x_1 - 1) + x_4}{1 + x_1^2 + x_2^2}
\]

(14)

This plant was the same as that used in Kim et al. [22]. In our model, only two input values, \( y_{p}(t) \) and \( u(t) \), were fed to the RFNFN model with SPSO learning algorithm to determine the output \( y_{p}(t) \). The training inputs were independent and identically distributed uniform sequence over \([-1, 1]\) for about half of the training time. A single sinusoid signal was given by \( 1.05 \sin(\pi t/45) \) for the remaining training time. There was no repetition on these 900 training data, that is, we had different training sets for each epoch. The checking input signal \( u(t) \) in the following equation was used to determine the identification results:

\[
u(t) = \begin{cases} 
\sin(\frac{\pi \cdot t}{25}) & 0 < t < 250 \\
1.0 & 250 \leq t < 500 \\
-1.0 & 500 \leq t < 750 \\
0.3 \sin(\frac{\pi \cdot t}{25}) + 0.1 \sin(\frac{\pi \cdot t}{32}) & 750 \leq t < 1000 \\
+ 0.6 \sin(\frac{\pi \cdot t}{10}) & 750 \leq t < 1000 
\end{cases}
\]

(15)

The SPSO learning algorithm is proceeded for 200 generations. And we set three fuzzy rules constitute the RFNFN model. The creation of three fuzzy rules, we are using the try-and-error methods to determine the number of fuzzy rules. They are given as follows:
Rule 1: IF $x_1$ is $u(-0.49976, -1.49337, -1.64946)$
and $x_2$ is $u(-1.63933, -1.24708, -0.84089)$
Then $\hat{y}_1 = 0.033959 - 0.18451 x_1 + 0.141187 \sin(\pi x_2) - 0.07847 \cos(\pi x_2)$

Rule 2: IF $x_1$ is $u(0.873601, 0.5818, -1.45324)$
and $x_2$ is $u(-0.80793, 1.07957, 1.420453)$
Then $\hat{y}_2 = 1.007304 - 0.982111 x_1 + 0.261444 \sin(\pi x_2) - 0.66635 \cos(\pi x_2)$

Rule 3: IF $x_1$ is $u(-1.18444, -0.3375, -0.27661)$
and $x_2$ is $u(-0.16341, 1.911395, -0.8027)$
Then $\hat{y}_3 = -1.15011 x_1 + 0.464354 \sin(\pi x_2) + 1.009515 \cos(\pi x_2)$
$+ 0.354662 x_1 + 0.090666 \sin(\pi x_2) + 0.39142 \cos(\pi x_2)$

where $u(m_j, \sigma_j, \theta_j)$ represents a Gaussian membership function with mean $m_j$, variance $\sigma_j$ and feedback weight $\theta_j$ in the $i$th input variable and the $j$th rule. After training of membership function on the two dimensions are shown in Fig. 8.

Figure 8. The distribution of the membership functions on the $x_1$ and $x_2$ dimensions.

In this example, we compared the proposed SPSO learning algorithm with the PSO and GA algorithms. Figure 9 shows the learning curves of the three methods. Figures 10(a)-(c) illustrate the identification outputs of the three methods. The errors between the desired output and model output are shown in Figs. 11(a)-(c). As shown in Figs. 10(a)-(c), the identification abilities of the SPSO algorithm were better than those of the PSO and GA methods. Therefore, we can find that the proposed SPSO algorithm converges quickly and obtains a lower rms error than PSO and GA.
Finally, the average, best and worst RMS errors after 200 generations are listed in Table 1. As shown in Table 1, we also find that the proposed SPSO method is better performance than other methods.

Table 1. Comparison of performance of SPSO, PSO, and GA in example 1.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SPSO</th>
<th>PSO</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.010369</td>
<td>0.017248</td>
<td>0.037759</td>
</tr>
<tr>
<td>Best</td>
<td>0.002766</td>
<td>0.004662</td>
<td>0.017624</td>
</tr>
<tr>
<td>Worst</td>
<td>0.021598</td>
<td>0.044778</td>
<td>0.089581</td>
</tr>
</tbody>
</table>

Example 2: Prediction of Chaotic Time Series

The Mackey-Glass chaotic time series \( x(t) \) was generated using the following delay differential equation:

\[
\frac{dx(t)}{dt} = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t).
\]

Crowder [21] extracted 1000 input-output data pairs \( \{x, y\} \) using four past values of \( x(t) \):

\[
[x(t - 18), x(t - 12), x(t - 6), x(t); x(t + 6)]
\]

where \( \tau = 17 \) and \( x(0) = 1.2 \). Four inputs to the RFNFN model, corresponded to these values of \( x(t) \), and one output was \( x(t+\Delta t) \), where \( \Delta t \) is a time interval into the future. The first 500 pairs (from \( x(1) \) to \( x(500) \)) were the training data set, while the remaining 500 pairs (from \( x(501) \) to \( x(1000) \)) were the testing data used to validate the proposed method. The evolution is proceeded for 200 generations. And we set three fuzzy rules constitute the RFNFN model. The creation of three fuzzy rules, we are using the try-and-error methods to determine the number of fuzzy rules. They are given as follows:

**Rule 1**: IF \( x_1 \) is \( u(1.883076, -4.78496, -1.92978) \)
and \( x_2 \) is \( u(-0.59831, -0.35093, 2.557971) \)
and \( x_3 \) is \( u(1.403306, 2.14404, 1.06741) \)
and \( x_4 \) is \( u(-0.80891, -0.47835, -0.0922) \)
Then \( \hat{y}_1 = 0.534532x_1 + 1.320135\sin(\pi x_1) + 0.052111\cos(\pi x_1) - 0.6761x_1 + 0.441931\sin(\pi x_1) + 0.464482\cos(\pi x_1) + 0.575191x_1 - 0.520323\sin(\pi x_1) + 0.271153\cos(\pi x_1) - 0.67443x_1 + 0.6619\sin(\pi x_1) - 0.57852\cos(\pi x_1) \)

where \( u(m_y, \sigma_y, \theta_y) \) represents a Gaussian membership function with mean \( m_y \), variance \( \sigma_y \) and feedback weight \( \theta_y \) in the \( i \)th input variable and the \( j \)th rule. After training of membership function on the four dimensions are shown in Fig. 12.

![Figure 12. The distribution of the membership functions on the \( x_1, x_2, x_3 \) and \( x_4 \) dimensions.](image)

In this example, as with example 1, we also compared the performance of the SPSO learning algorithm with other methods. Figure 13 shows the learning curves of the three methods. Figures 14(a)-(c) illustrate the prediction outputs of the various models. The errors between the desired output and model output are shown in Figs. 15(a)-(c) As shown in Figs. 13-15, the prediction abilities of the proposed SPSO method were better than those of the PSO and GA methods. The comparison results of rms error are tabulated in Table 2. The comparison results indicate that the predicting RMS error for the SPSO method are better than those obtained using other methods.

![Figure 13. Learning curves the proposed SPSO, the PSO and the GA methods.](image)
5. Conclusions

This paper proposed a recurrent functional neural fuzzy network (RFNFN) with SPSO for solving the identification and prediction problems. The recurrent networks have feedback connections in their topologies, and these feedback loops are used to memorize past information. So, they can be used to deal with temporal problems in prediction and identification. An efficient learning algorithm, called symbiotic particle swarm optimization (SPSO), combined the symbiotic evolution and new velocity update function of the modified particle swarm optimization. The proposed learning algorithm can find the global optimal quickly and the good performance more efficiently. Experiment results demonstrated that the converging speed and RMS error of the proposed method is better than other methods.

References

C.-J. Lin et al.: An Efficient Symbiotic Particle Swarm Optimization for Recurrent Functional Neural Fuzzy Network Design


**Cheng-Jian Lin** received the B.S. degree in electrical engineering from Ta-Tung University, Taiwan, R.O.C., in 1986 and the M.S. and Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Taiwan, R.O.C., in 1991 and 1996. Currently, he is a full Professor of Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taiwan, R.O.C. His current research interests are soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and FPGA design. Dr. Lin is a member of the Phi Tau Phi. He is also a member of the Chinese Fuzzy Systems Association (CFSA), the Chinese Automation Association, the Taiwanese Association for Artificial Intelligence (TAAI), the IEEE Systems, Man, and Cybernetics Society, and the IEEE Computational Intelligence Society. He is an executive committee member of the Taiwanese Association for Artificial Intelligence (TAAI) and the Chinese Fuzzy Systems Association (CFSA).

**Chi-Feng Wu** received the B.S. degree in atmospheric sciences from Taiwan University, Taiwan, R.O.C., in 1991 and the M.S. degree in information and computer education from Taiwan Normal University, Taiwan, R.O.C., in 1994. Currently, he is a Lecturer of Information Management and Communication Department, Wenzao Ursuline College of Languages, Taiwan, R.O.C. His current research interests are image processing and neural fuzzy networks.