6

Recurrent Fuzzy Neural Networks and Their Performance Analysis

R.A. Aliev[1], B. Fazlollahi[2], B.G. Guirimov[3] and R.R. Aliev[4]
[1,3]*Azerbaijan,*
[2]*USA,*
[4]*North Cyprus*

1. Introduction

There are many papers that consider different structure and training algorithms of FNN. Within their structural range, the networks may differ by type of signals (singleton, interval, general fuzzy, triangle shaped or other), topology (layered, fully-connected, with or without feed-back connections, feed-back connections in all or some of the layers etc), type of neurons (transfer function, same type in all layers or different depending on layer).
Note also that FNN is further complicated when we deal with applications of temporal character such as dynamic control, forecasting, identification, recognition of temporal sequences (e.g. voice recognition). It is obvious that in this case classical FNN with feed-forward structure, operable mainly for memory-less problems, would be ineffective. In this respect there is a strong demand for recurrent fuzzy neural networks (RFNN) with dynamic mapping capability, temporal information storage, dynamic fuzzy inference, and as a result, capable of solving temporal problems [7,24,27].
Paper [48] discusses delay feedback neuro-fuzzy networks and their usability to effectively tackle dynamic systems. This is a simplified version of recurrent network with feedback connections at only one layer of the network. To train unknown parameters of RFNN the author of [1] uses a supervised learning algorithm that requires differentiability of the membership functions that is not always possible.
In [25] a recurrent self-organizing neuro-fuzzy inference network is proposed. The main characteristic of this system is the ability to deal with temporal problems including dynamic fuzzy inference. The system with on-line learning feature is capable also of building the structure and (crisp) parameters of the network. The learning algorithm is based on the use of the ordered derivative (partial derivative) produced with the use of an ordered set of equations. The efficiency of the proposed neuro-fuzzy system is verified on the basis of various simulations on benchmark temporal problems, including time-sequence prediction, adaptive noise cancellation, dynamic plant identification, and non-linear plant control.
In [27] a recurrent multi-layered connectionist network for realizing fuzzy inference using dynamic fuzzy rules is presented. The paper distinguishes as containing good methodological support encompassing important aspects of neuro-fuzzy systems class. The back-propagation algorithm is used as the learning algorithm minimizing the cost function to achieve necessary connection weights and biases. As in [25], several examples and

Source: Recurrent Neural Networks, Book edited by: Xiaolin Hu and P. Balasubramaniam, ISBN 978-953-7619-08-4, pp. 400, September 2008, I-Tech, Vienna, Austria

performance comparisons with the existing works are presented including time sequence prediction, identification of non-linear dynamic system, identification of a chaotic system, and adaptive control of a non-linear system.

It should be noted that in [27] feedback links in the second layer only are added to the fuzzy feed-forward neural network. This rather simplified version of neuro-fuzzy network has crisp feed-forward connection weights and non-adjustable recurrent connection weights in the second layer. These simplifications undoubtedly lead to some decrease in the efficiency of the proposed neuro-fuzzy network.

In [35] a dynamic neuro-fuzzy system consisting of recurrent TSK rules is investigated. The suggested network is trained by dynamic fuzzy neural constrained optimization method based on the concept of constrained optimization. The proposed dynamic neuro-fuzzy system is tested on two temporal examples and the noise cancellation problem.

In [31] a hybrid supervisory control system using a recurrent neuro-fuzzy network, with the network output feeding back to the network input through time-delay units, is proposed. An on-line training methodology which is based on Lyapunov stability theorem and the gradient descent method is proposed. Some simulated and experimental results are provided to demonstrate the efficiency of the proposed neuro-fuzzy system.

Recurrent neuro-fuzzy systems for implementation of long-range prediction fuzzy model is investigated in [54]. In this recurrent neuro-fuzzy model the network output is fed back to the network input through one or more time delay units. Levenberg-Marquardt algorithm with regularization is used for adjusting crisp weights and biases of the feed-forward and feed-back connections of the recurrent neuro-fuzzy network. The suggested neuro-fuzzy network is applied to modeling and control of a neutralization process.

In [50] a direct adaptive iterative learning control system based recurrent neuro-fuzzy network is presented. The analysis of stability and learning is studied. A computer simulation for an inverted pendulum system and Chua's chaotic circuit is demonstrated.

A sliding mode recurrent neuro-fuzzy network based control system is proposed in [30] to control the mover of a permanent-magnet linear synchronous motor. The learning algorithm used is the same as in [31].

Interesting design methods and applications of FRNN are discussed in [18,26,29,34,55]. In paper [34] a discrete mathematical model of RFNN is constructed and a learning algorithm adopting a recursive least square approach is used to identify the unknown parameters in the model. In [18] the authors propose an efficient algorithm for determination of structure of model and identification of its parameters with the aim of producing improved predictive performance for NARMAX (nonlinear autoregressive moving average with exogenous inputs) time series models. A fuzzified TSK (Takagi-Sugeno-Kang) type recurrent fuzzy network is developed in paper [26] for one-dimensional and two-dimensional fuzzy temporal sequence prediction. Paper [29] considers a design method of recurrent fuzzy neural network based adaptive hybrid control for multi-input multi-output linearized dynamic systems. The proposed control system is applied to aircraft flight control system. Paper [55] deals with adaptive nonlinear noise control systems using recurrent fuzzy neural networks, the feedback connections of which are used to create dynamic fuzzy rules trained using dynamic back-propagation learning algorithm. The learning of fuzzy weights of FRNN is not considered in these works as all the papers assume network weights to be crisp numbers.

In [20] a self-organizing adaptive fuzzy neural network for nonlinear systems is proposed. The identifier is used to estimate the controlled system's dynamic with the learning of fuzzy neural network. The parameter learning algorithms are derived based on Lyapunov function candidate.

A very important role in designing fuzzy neural networks takes its learning method and the problem of how to train fuzzy neural networks (FNN) has great scientific and practical interest and is becoming challenging and important research area.

The training methods for neural networks can be divided into two large categories: gradient-based algorithms and evolutionary algorithms. The overview of the works on training methods for fuzzy feed-forward neural networks is given in [10]. Work [32] needs special note as presenting some methodological support from the viewpoint of fuzzy neural networks. Paper [32] develops two learning algorithms for fuzzy feed-forward neural networks that is the fuzzy back-propagation algorithm and the fuzzy conjugate gradient (CG) algorithm for determination of fuzzy weights and biases represented as Π-type fuzzy numbers. The authors use GA for determination of optimal learning rate at each iteration step of fuzzy CG algorithm. Some real simulations realizing non-dynamic fuzzy inference rules and fuzzy functions are demonstrated.

The evolutionary algorithms based approach to training of FNN involves application of genetic algorithms and other population-based natural evolution inspired algorithms to minimize error function and determine the fuzzy connection weights and biases [10,28]. In contrast to BP and other supervised learning algorithms, evolutionary algorithms do not use the derivative information, and hence, they are most effective in case where the derivative is very difficult to obtain or even unavailable. Moreover, the calculation complexity of BP algorithms is high due to the need for computing complex Hessian or Jacobian matrices.

In [47] nonlinear neural network predictive control strategy based on chaotic particle swarm optimization is presented. It is shown that since the back-propagation algorithm is easily trapped in local minima and its convergence performance greatly depends on its learning rate parameter and initial conditions, the weights and biases of the neural network are optimized by particle swarm optimization algorithm. Learning of crisp weights and biases is considered in this work.

TSK-type recurrent neuro-fuzzy system trained by GA is proposed in [24]. In this network internal variables, derived from fuzzy firing strengths are fed back to both network input and output layers. To train the proposed TSK-type recurrent neuro-fuzzy network, a GA based method is developed. The recurrent neuro-fuzzy network with genetic learning is applied to dynamic system control problem. The research in this field is at its infancy and many fundamental problems such as choosing the most efficient error function, coding technique, and genetic strategies remain to be solved [33].

Unfortunately, little progress has been made in the development of recurrent fuzzy neural networks processing directly fuzzy information and using fuzzy weights and biases as adjustable parameters. For the first time some attempts were made in [5,6,8-10,22] to develop an efficient RFNN with fuzzy inputs, fuzzy weights expressed as fuzzy numbers, and fuzzy outputs. In this study we consider the structure, operation, and DE-based training algorithm for multi-layer recurrent fuzzy neural network processing fuzzy signals and demonstrate its efficiency on a number of benchmark and application problems.

The rest of this paper is organized as follows. In section 2 we cover prerequisite material (such as fuzzy function, Hamming distance, fuzzy neural networks, differential evolution optimization, etc.) to be used in the study. Section 3 formulates the statement of problem of creating RFNN with efficient learning algorithm. Section 4 illustrates the structure and computational procedure of the investigated recurrent fuzzy neural network. In section 5 the recurrent fuzzy neural network learning algorithm using DEO is described. Simulations and experimental results are discussed in section 6. Section 7 gives the conclusion of this paper.

2. Preliminaries

In this section, we briefly review some prerequisite material which will be of help in the development of the concepts of evolutionary computing based learning of RFNN. While the reader may find some of the definitions in the literature, we augment them with some interpretation which could be useful in the context of our considerations.

2.1 Fuzzy function

Briefly speaking, by a fuzzy function we mean a function, whose values are fuzzy numbers. Let *f* be a fuzzy function, $\mu_{f(x)}$ denotes the membership function of the fuzzy number $f(x)$, and for $0<\alpha\le 1$, $f_+^\alpha(x)$ will denote $\sup\{z\in dom(\mu_{f(x)}):\ \mu_{f(x)}(z)\ge\alpha\ \}$ and $f_-^\alpha(x)$ will denote $\inf\{z\in dom(\mu_{f(x)}):\mu_{f(x)}(z)\ge\alpha\ \}$. Functions $f_-^\alpha(x)$ and $f_+^\alpha(x)$ are level functions of *f*.

A fuzzy subset *A* of R^n is defined in terms of its membership function $\mu_A(x):R^n\to[0,1]$ For each $\alpha\in(0,1]$ the α -level set $[\mu_A(x)]^\alpha$ of a fuzzy set *A* is the subset of points $x\in R^n$ with membership values $\mu_A(x)$ of at least α, that is $[\mu_A(x)]^\alpha=\{x\in R^n:\ \mu_A(x)\ge\alpha\}$.

2.2 Distance

Formally, the distance *d(x,y)* between *x* and *y* in R^n is considered to be a two-argument function satisfying the conditions: *d(x,y)* ≥ 0, for every *x* and *y*; *d(x,x)*=0, for every *x*; $d(x,z)\ge d(x,y)+d(y,z)$ for every pattern *x*, *y* and *z*. In the case of continuous variables we have a long list of distance functions [7, 40].

Let us consider the space E^n of all fuzzy subsets of R^n which satisfy the conditions of normality, convexity and are upper semicontinuous with compact supports $[\mu_A(x)]^{\alpha=0}$. For fuzzy sets *A* and *B* in E^n, in general, the Minkowski distance defined as follows

$$d(A,B)=\sqrt[p]{\int_X|\mu_A(x)-\mu_B(x)|^p\,dx},\ p\ge1,\tag{1}$$

where **X** - is a universe of discourse. This distance satisfies the above mentioned conditions. In particular, when $p=1$ we get Hamming distance.

2.3 Fuzzy neural networks and neuro-fuzzy systems

Fuzzy neural network (FNN) approach has become a powerful tool for solving real-world problems in the area of forecasting, identification, control, image recognition and others that are associated with high level of uncertainty [2,7,10,11,14,23,24,23]. This is related with the fact that the FNN paradigm combines the capability of fuzzy reasoning in handling uncertain information and the capability of pure neural networks in learning from experiments [48]. An advantage of FNN is that it allows automation of design of fuzzy rules and combined learning of numerical data as well as expert knowledge expressed as fuzzy IF-THEN rules [7]. FNN may have smaller network size and be faster in convergence speed as compared with ordinary NN.

There are two different approaches in academic literature. First approach is neuro-fuzzy systems whose main task is to process numerical relationships [27]. Many papers, including papers [31,37,48] combine features of neural and fuzzy approaches into Neuro-Fuzzy systems. Second approach is fuzzy neural systems with the objective to process both numerical (measurement based) information and perception based information. The FNN of this (second) class are oriented for real-world problems that are inherently uncertain and imprecise [10,19,32]. It is necessary to point out that neuro-fuzzy systems cannot replace fuzzy neural systems because unlike the former which perform mapping from non-fuzzy input signals to non-fuzzy outputs, the latter process linguistic information directly.

When we deal with linguistic information, i.e. work at a higher data abstraction level, we should employ fuzzy neural networks, not neuro-fuzzy networks, to solve the considered problem approximately [21].

2.4 Differential evolution optimization method

Recently many heuristic algorithms have been proposed for global optimization of nonlinear, non-convex, and non-differential functions [3,12,41,51]. These methods are more flexible than classical as they do not require differentiability, continuity, or other properties to hold for optimizing functions. Some of such methods are genetic algorithm, evolutionary strategy, particle swarm optimization, and differential evolution (DE) optimization. In this study we consider the use of the DE algorithm.

As a stochastic method, DE algorithm uses initial population randomly generated by uniform distribution, differential mutation, probability crossover, and selection operators [42]. The population with *ps* individuals are maintained with each generation. A new vector is generated by mutation which in this case is randomly selecting from the population 3 individuals: $r_1 \neq r_2 \neq r_3$ and adding a weighted difference vector between two individuals to a third individual (population member).

The mutated vector is then undergone crossover operation with another vector generating new offspring vector.

The selection process is done as follows. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector will replace the vector with which it was compared in the following generation.

Extracting distance and direction information from the population to generate random deviations results in an adaptive scheme with excellent convergence properties. DE has been successfully applied to solve a wide range of problems such as image classification, clustering, optimization etc.

Figure 1 shows the process of generation new trial solution vector from randomly selected population members. Here we assume that the solution vectors are of dimension 2 (i.e. 2 optimization parameters).

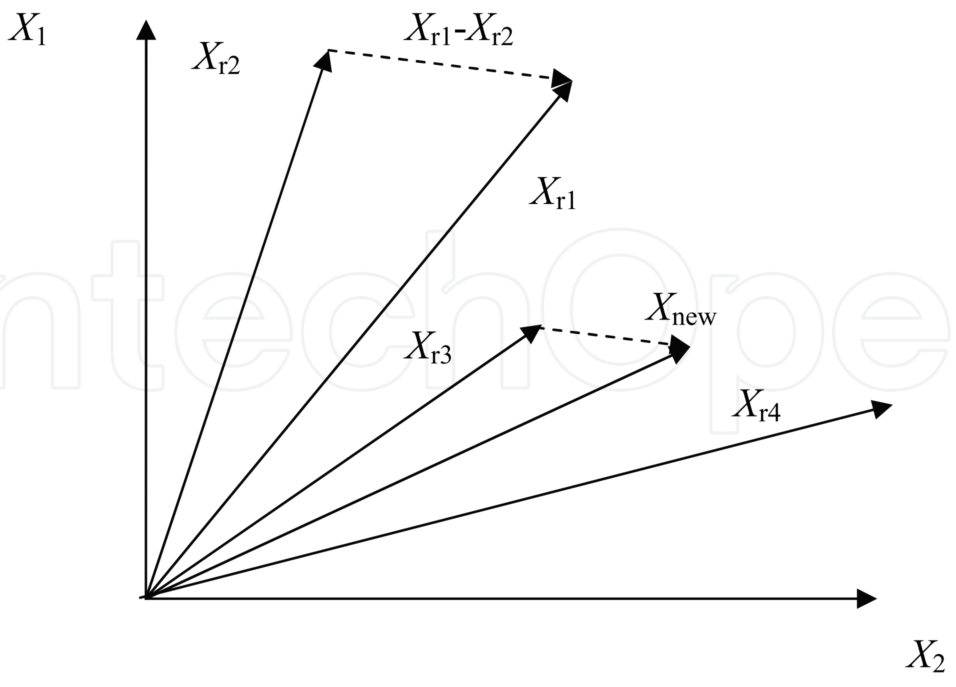

Figure 1

3. Statement of problem

Assume that an unknown nonlinear system is expressed as follows [4]:

$$\tilde{y}(t) = \tilde{g}\big(\tilde{y}(t-1),...,\tilde{y}(t-n),\tilde{u}(t-1),...,\tilde{u}(t-m)\big), \tag{2}$$

where $\tilde{y}(t)$ and $\tilde{u}(t)$ are the output and input of the system, respectively, represented as fuzzy valued function, $\tilde{g}(.)$ is an unknown nonlinear fuzzy mapping to be estimated by RFNN, n and m are order of the system. It is required to design RFNN such that its output $\tilde{y}_N(t)$ determined as

$$\tilde{y}_N(t) = \tilde{g}_N\left(\tilde{y}(t-1),\tilde{u}(t-1),\tilde{W},\tilde{V},\tilde{\theta}\right), \tag{3}$$

will be as close as possible to $\tilde{y}(t)$ (1), where $\tilde{W},\tilde{V},\tilde{\theta}$ collectively define the structure and set of parameters of RFNN: forward connection weights, backward (recurrent) connection weights, and biases, respectively.
As measure of closeness between $\tilde{y}(t)$ and $\tilde{y}_N(t)$ we need to define a suitable error function serving as a distance (metric). For continuous variables there is a long list of distance functions [4,40]. In this paper we will use the well-known and commonly used Hamming distance. Therefore the problem of learning of FRNN is an optimization problem with the purpose of adjusting fuzzy parameters $\tilde{W} = \{\tilde{w}_{lij}\}$, $\tilde{V} = \{\tilde{v}_{lij}\}$, and $\tilde{\theta} = \{\tilde{\theta}_{li}\}$ to minimize the error function

$$\widetilde{E} = \sum\sum\left|\widetilde{y}_{pi} - \widetilde{y}_{Npi}\right|, \tag{4}$$

where $\widetilde{y}_{pi}$ is the desired value and $\widetilde{y}_{Npi}$ is the actual value of RFNN output layer's neuron *i* when applied training patter *p*, $\widetilde{E}$ is hamming distance.

Training algorithm is critical to RFNN as it will affect RFNN approximation capability. Due to the type of error function, we cannot use here the BP algorithm. Another problem with the BP is that it is easily trapped in local minimima and its convergence performance greatly depends on its learning rate parameter and the initial conditions. As optimization strategy for training RFNN we will use an evolutionary computing strategy, namely, DEO method.

During the training, the weights of feed-forward and feed-back connections and biases of RFNN are optimized by the differential evolution algorithm which would lead to the minimum of error function (4).

It is worth to note that using clustering based differential evolution algorithm for training of RFNN may give a higher performance [51]. Such an approach will be used on considering a petrol production forecasting example.

4. Recurrent fuzzy neural network structure and computation

The general structure of a recurrent fuzzy neural network is presented in Figure 2. The box elements represent memory cells that store values of activation of neurons at previous time step, which is fed back to the input at the next time step.

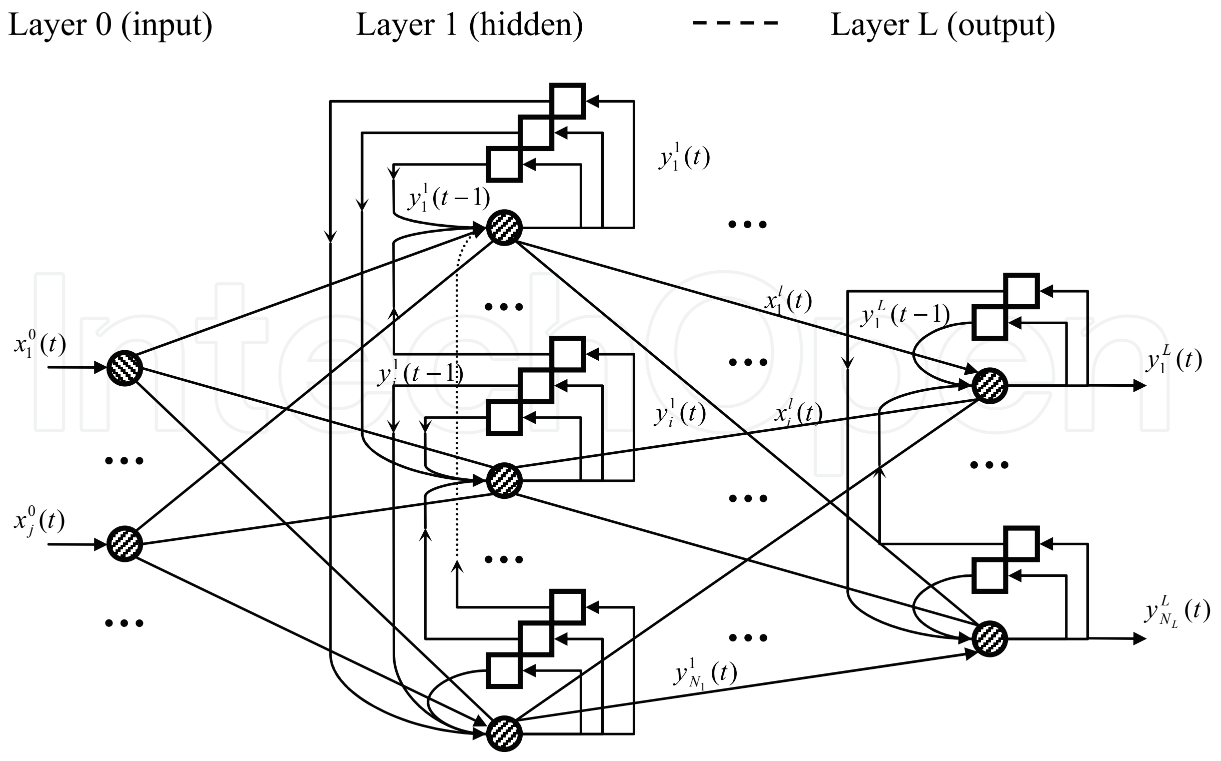

Figure 2. The structure of RFNN

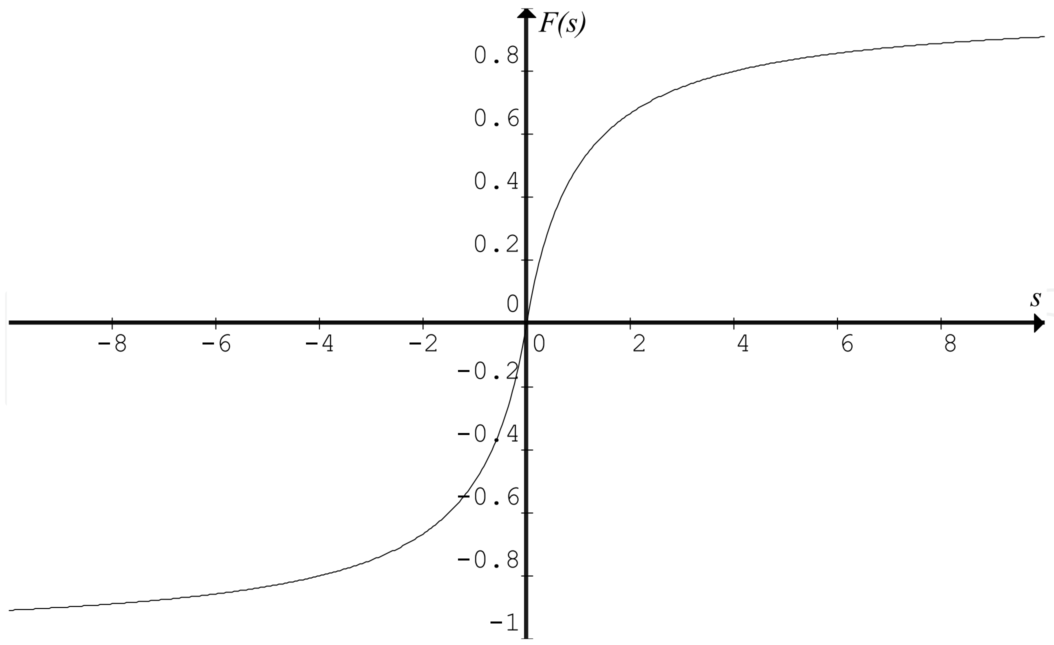

Figure 3. The activation function *F*(*s*)

In general, the network may have virtually any number of layers. We number the layers successively from 0 (the first or input layer) to *L* (last or output layer). The neurons in the input layer (layer 0) only distribute the input signals without modifying their values.

$$\tilde{y}_i^0(t) = \tilde{x}_i^0(t) \tag{5}$$

The neurons in the remaining layers (layer 1 to layer *L*-1) are dynamic and compute their output signals as follows:

$$\tilde{y}_i^l(t) = F\left(\tilde{\theta}_i^l + \sum_j \tilde{x}_j^l(t)\tilde{w}_{ij}^l + \sum_j \tilde{y}_j^l(t-1)\tilde{v}_{ij}^l\right), \tag{6}$$

where $\tilde{x}_j^l(t)$ is *j*-th fuzzy input to the neuron *i* at layer *l* at the time step *t*, $\tilde{y}_i^l(t)$ is the computed output signal of the neuron at the time step *t*, $\tilde{w}_{ij}$ is the fuzzy weight of the connection to neuron *i* from neuron *j* located at the previous layer, $\tilde{\theta}_i$ is the fuzzy bias of neuron *i*, and $\tilde{y}_j^l(t-1)$ is the activation of neuron *j* at the time step (*t*-1), $\tilde{v}_{ij}$ is the recurrent connection weight to neuron *i* from neuron *j* at the same layer.
The neurons at the last layer (layer *L*) are linear and their outputs are calculated as:

$$y_i^L(t) = \theta_i^L + \sum_j x_j^L(t)w_{ij}^L + \sum_j y_j^L(t-1)v_{ij}^L \tag{7}$$

The activation *F* for a total input to the neuron *s* (figure 3) is calculated as:

$$F(s) = \frac{s}{1+|s|} \tag{8}$$

So, the output of neuron *i* at layer *l* ($l = \overline{1, L-1}$) is calculated as follows:

$$y_i^l(t) = \frac{\tilde{\theta}_i^l + \sum_j \tilde{x}_j^l(t)\tilde{w}_{ij}^l + \sum_j \tilde{y}_j^l(t-1)\tilde{v}_{ij}^l}{1 + \left|\tilde{\theta}_i^l + \sum_j \tilde{x}_j^l(t)\tilde{w}_{ij}^l + \sum_j \tilde{y}_j^l(t-1)\tilde{v}_{ij}^l\right|} \tag{9}$$

The total number of connections (including forward, recurrent, and biases) is equal to $(N_I + 1)N_H + (N_H + 1)N_O + N_H^2 + N_O^2$,
where N_I is number of inputs (here we use 1 input for simplicity), N_H is the number of neurons in the second (hidden) layer of RFNN and N_O is the number of outputs. Thus, for a network with 1 input, 3 hidden neurons, and 1 output, there will be 20 connections overall. Every forward connection weight, recurrent connection weight, and bias value are represented as a triangular fuzzy number [16,17,37-39]: $\tilde{w}_{ij}^l = T(w_{Lij}^l, w_{Aij}^l, w_{Rij}^l)$, $\tilde{v}_{ij}^l = T(v_{Lij}^l, v_{Aij}^l, v_{Rij}^l)$, $\tilde{\theta}_i^l = T(\theta_{Li}^l, \theta_{Ai}^l, \theta_{Ri}^l)$, respectively. Note that the forward connection is from neuron j at layer (*l*-1) to neuron *i* at layer *l*, while the recurrent connections are between the outputs and inputs of the neurons at the same layer. Triangle fuzzy numbers are described as *T*(*a*,*b*,*c*), where [*a*,*c*] is the fuzzy number support and *b* is the value with membership equal to 1 (the average value).

Inputs to RFNN x_i can accept system outputs at previous stages, $x_1 = y_N(t-1),\ x_2 = y_N(t-2), ..., x_r = y_N(t-r)$, etc., as well as exogenous signals $x_{r+1} = u(t),\ x_{r+2} = u(t-1),\ x_{r+3} = u(t-2), ...$

For example, for constructing a RFNN based time series predictor ($\hat{F}_{NN}$) $\hat{y}_{t+1} = \hat{F}_{NN}(y_t, y_{t-1}, ..., y_{t-n+1}, u_t, ..., u_{t-1}, u_{t-m+1})$, where y_t is the value of time series data at time interval *t*, u_t is the value of an additional (second) factor at time interval *t*, the above presented structure could be modified as given in Figure 4.

In case the original learning patterns are crisp, we need to sample data into fuzzy terms, i.e. to fuzzify the learning patterns. The fuzzifiers can be created independently for specific problems. A different approach that is used in this paper is to convert the numeric data into information granules by fuzzy clustering [40]. In this case the receptive fields forming the input layer of RFNN are constructed using clustering. Fuzzy clusters fully reflect the character of the data.

5. Differential evolution optimization based learning of RFNN

The considered RFNN requires the global parameter optimization method suitable for nonlinear, non-convex, and non-differentiable mapping functions. Ideally, we want to find the global minimum of (4) and this requires more careful selection of the optimization engine. Despite the fact that the gradient descent based methods are predominant, they are not global optimizers. Most suitable are population based optimization techniques including genetic algorithms, evolutionary strategy, particle swarm optimization, DEO, etc. In this paper we use DEO method which has many advantages over other evolutionary algorithms and GA [3,42,51]. There are some reasons for using DEO in RFNN learning problem. First, DEO supports a search mechanism of global nature. DEO is useful when dealing with different distance functions including Hamming distance, Tschebyshev distance (gradient descent based methods require distance functions be differentiable, e.g. Euclidean distance function).

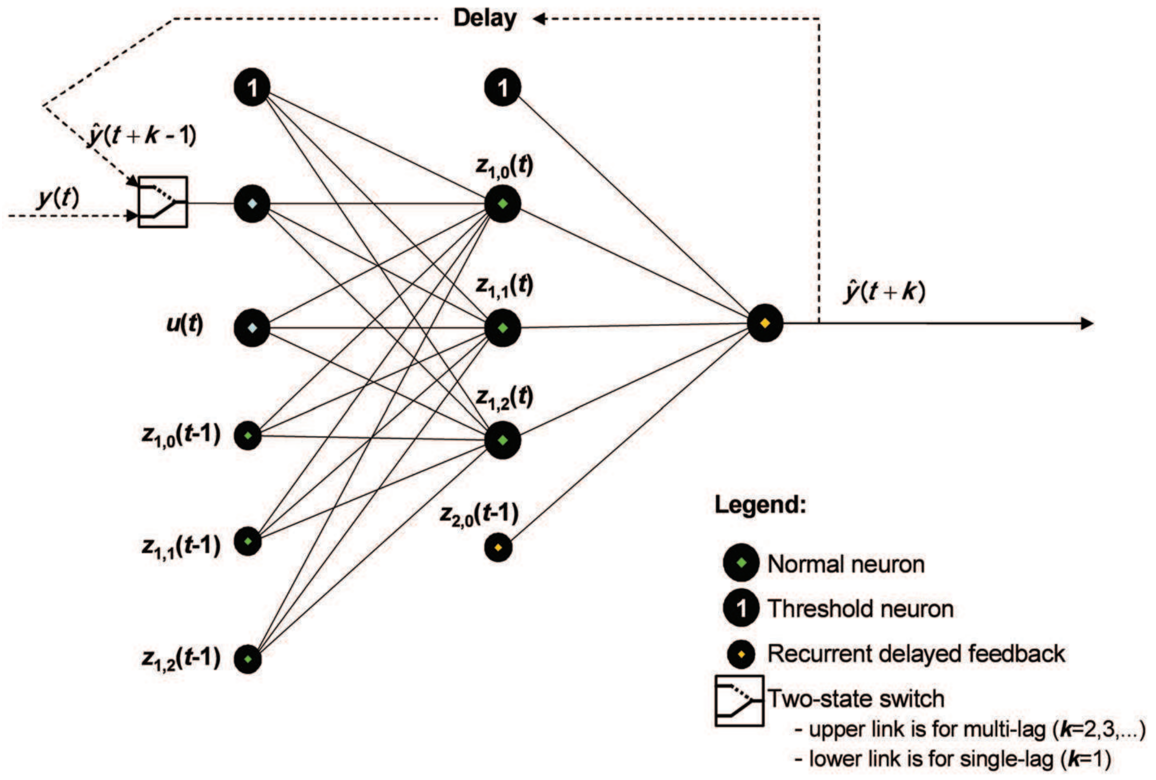

Figure 4. The structure of a simple FRNN

For application of an evolutionary algorithm for learning RFNN we consider the population individ to represent a whole combination of weights ($\widetilde{W} = \{\widetilde{w}_{lij}\}$, $\widetilde{V} = \{\widetilde{v}_{lij}\}$) and biases ($\widetilde{\theta} = \{\widetilde{\theta}_{li}\}$) (i.e. parameters of RFNN) defining the input/output mapping (3). The population maintains a number of popential parameter sets defining different RFNN solutions and recognizes one of these solutions to be the best solution. This best solution is the one with minimum training error. After a series of generations, the best solution may converge to a near-optimum solution, which would represent in our case a RFNN with the required accuracy.

To apply an evolutionary population based optimization algorithm we first should identify the optimized parameter vector. For training RFNN we need to optimize values of: forward connection weights, processing (hidden and output) neuron biases, and recurrent weights. According to the structure of RFNN given in section 2, the number of all parameters to be adjusted during the learning process and therefore the dimension of the optimized parameter vector (for a FRNN with one hidden layer) is

$$N_{par} = (N_I + 1)N_H + (N_H + 1)N_O + N_H^2 + N_O^2. \qquad (10)$$

Before starting training all the parameters are initialized by randomly chosen values, usually not beyond the interval [-1,1]. This constraint is further enforced to the parameters associated to backward connections. It means that during further training steps the values of forward weights and biases can go beyond the interval [-1,1] while the values of backward connection are kept within this interval. This additional constraint is added to make RFNN stable which means that under the constant input the value of output will converge to a constant value (either crisp or fuzzy).

Prior launching the optimization we set parameter *f* of DEO to a positive value (typically about 0.9), define the DEO cost function to be the RFNN error function (4), and choose the population size (typically ten times the number of optimization parameters, i.e. $10N_{par}$).

Then the differential evolution optimization is started.

DEO based RFNN training algorithm can be summarized as follows:

Step 0. Initialize DE

- Step 0.0 Define the structure of RFNN: N_i, N_h, N_o
- Step 0.1. Construct template parameter vector X of dimension N_{par} according (10) for holding RFNN weights and biases: $X=\{\tilde{W},\tilde{V},\tilde{\theta}\}$
- Step 0.2. Set algorithm parameters: *f* (mutation rate), *cr* (crossover rate), and *ps* (size of population)
- Step 0.3. Define the cost function as function of error function of current RFNN parameters: $\tilde{E} = \sum\sum\left|\tilde{y}_{pi} - \tilde{y}_{Npi}\right|$

Step 1. Randomly generate *ps* parameter vectors (from respective parameter spaces (e.g. in the range [-1, 1]) and form a population $P=\{X_1, X_2, ..., X_{ps}\}$

Step 2. While Termination condition (number of predefined generations reached or required error level obtained) is not met generate new parameter sets:

- Step 2.1. Choose a next vector X_i (i=1,...,ps)
- Step 2.2. Choose randomly different 3 vectors from *P*: X_{r1}, X_{r2}, X_{r3} each of which is different from current X_i
- Step 2.3. Generate trial vector $X_t=X_{r1}+f(X_{r2}-X_{r3})$
- Step 2.4. Generate new vector from trial vector X_t. Individual vector parameters of X_t are inherited with probability *cr* into the new vector X_{new}. If the cost function from X_{new} is better (lower) than the cost function from X_i, current X_i is replaced in population *P* by X_{new}

  Next i

Step 3. Select the parameter vector X_{best} (RFNN parameter set) with best cost (training error $\tilde{E}$) function from population *P*. Extract from X_{best} vectors $\tilde{W},\tilde{V},\tilde{\theta}$ defining weights and thresholds for RFNN

Step 4. Stop the algorithm

If the obtained total error performance index or the behavior of the obtained network is not desired, we can restructure the network by adding new hidden neurons, or do better granulation of the learning patterns.

During the DE optimization process the solutions resulting in lower cost values have more chances to survive and be saved into a new population for participation in future generations. The process is repeated iteratively. During succeeding generations we keep into the population the solution that produced the lowest value of cost function of all previous generations. The farther we go with generations the higher is the chance to find a better solution.

6. Experiments and application

In this section, we report on the results of simulation of the suggested RFNN with DEO learning and compare the performance of RFNN and existing approaches. The performance of the proposed algorithm is examined on 3 benchmark problems in literature [13,27,46].

6.1 Non-linear system identification

We start with non-linear system studied in [13,28,32,45] as a benchmark identification problem.
The dynamic system is described by the equation:

$$y(k)=g(y(k-1),\, y(k-2))+u(k) \tag{11}$$

where:

$$g(y(k-1), y(k-2)) = \frac{y(k-1)y(k-2)(y(k-1)-0.5)}{1+y^2(k-1)+y^2(k-2)} \tag{12}$$

The system output depends on both its past values and current input. The goal is to approximate the model (11)-(12) by RFNN.
The RFNN for this example has 2 input neurons, 6 neurons at layer 1 and one output neuron. The number of all connections (including forward, backward, and biases) was 62. On the basis of (12) 400 data were created using random (in interval [-1,1] signal u and used for training. The trained network was tested on the basis of 200 test data created using (12) by applying sinusoidal signal $u = \sin(2\pi k / 25)$.
DE Optimization progress (MSE vs. successful iterations) is shown in Figure 5.
Table 1 shows a fragment of results (reached MSE) from intensive simulation experiments.

Experiment	MSE on train data	MSE on test data
1	0.0000818477	0.000342153
2	0.000116419	0.000738948
3	0.0000761331	0.000361591
4	0.0000235756	0.0000656312
5	0.000373168	0.00105125
6	0.000103995	0.000215761
7	0.0000696317	0.000204256

Table 1. RFNN training simulations for non-linear system identification

The final reached MSE at the best experiment was 0.000024 on training data and 0.000066 on test data. Table 2 presents comparative results obtained by different methods given in literature.

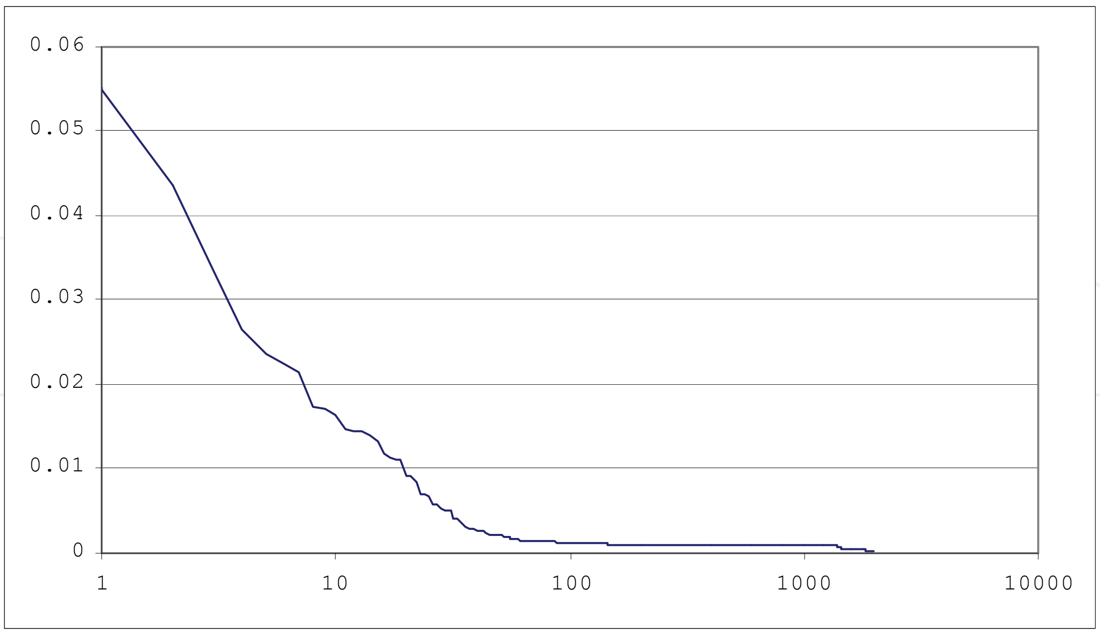

Figure 5. RFNN error convergence

Reference fuzzy model	MSE on training set	MSE on test set
[53]	-	0.00080
[45]	0.00075	0.00035
[13]	0.00010	0.00032
RFNN (our approach)	0.000024	0.000066

Table 2. Comparative results by different methods

The comparison of the actual and identified curves for y(k) is illustrated in Figure 6.

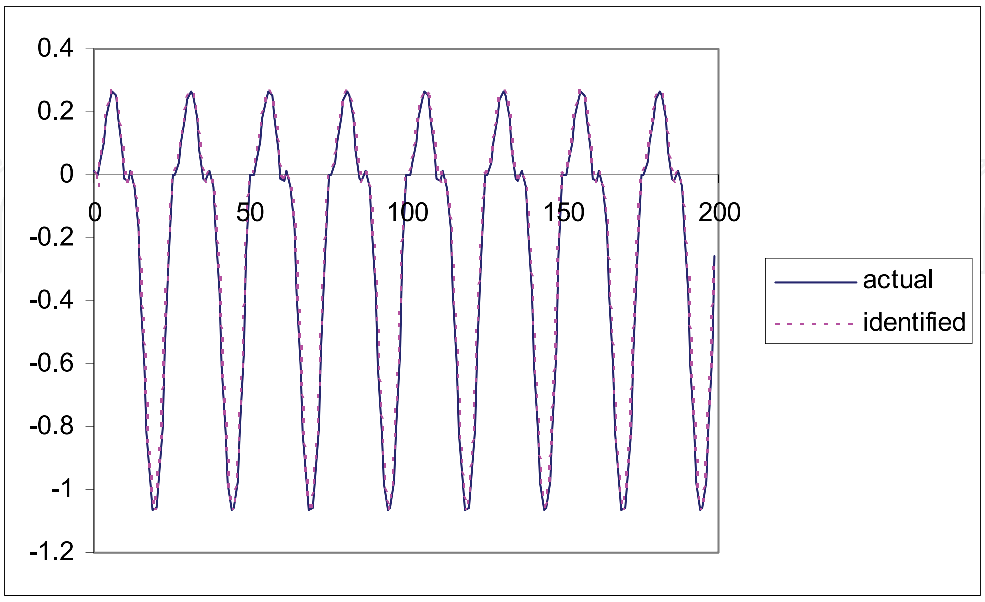

Figure 6. RFNN identification performance

6.2 Dynamic plant identification

This example is taken from [25,27] in which a nonlinear plant with multiple time-delay is identified. The nonlinear plant is described as follows:

$$y_p(k+1) = f(y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)) \tag{13}$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2} \tag{14}$$

In this example the output depends on three previous outputs and two previous inputs. For better results 2000 data were used for training generated by applying random $u(k)$ in interval [-1, 1].

For the testing signal $u(k)$ the following equation was used:

$$u(k) = \begin{cases} \sin\left(\frac{\pi k}{25}\right), & 0 \le k < 250 \\ 1, & 250 \le k < 500 \\ -1, & 500 \le k < 750 \\ 0.3\sin\left(\frac{\pi k}{25}\right) + 0.1\sin\left(\frac{\pi k}{32}\right) + 0.6\sin\left(\frac{\pi k}{10}\right), & 750 \le k < 1000 \end{cases} \tag{15}$$

Figure 7 shows the comparison of the desired test and RFNN output curves of the considered dynamic system.

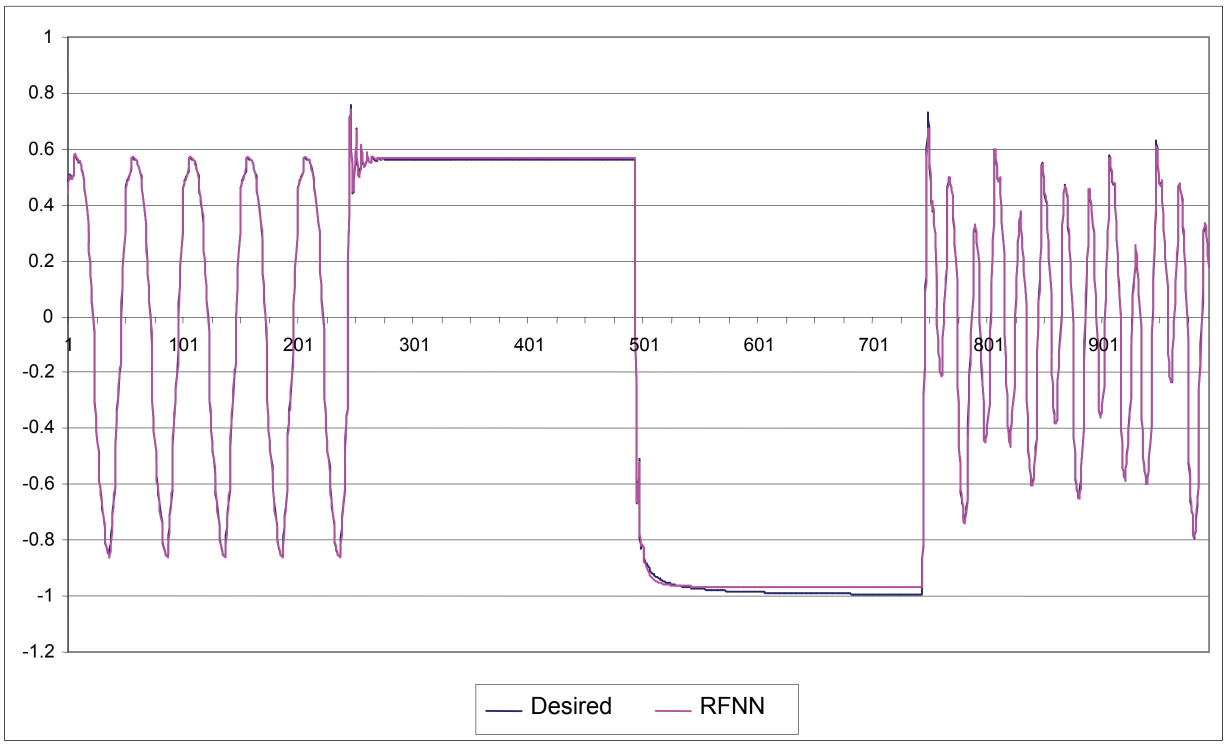

Figure 7. Comparison of output of RFNN with the desired output

The network used two input neurons ($y(k)$ and $u(t)$, respectively), 8 hidden neurons, and one output neuron ($y(k+1)$). In case of a non-recurrent FNN we would need 5 input neurons ($y(k)$, $y(k\text{-}1)$, $y(k\text{-}2)$, $u(t\text{-}1)$) and 1 output neuron ($y(k+1)$). In comparison with a regular FNN, the use of RFNN allows significant simplification of the network structure. Table 3 below shows the comparison of characteristics of fuzzy neural networks suggested in [27] and our RFNN.

	RFNN [27]	FNN [27]	RFNN (our approach)
No of inputs	2	5	2
No of outputs	1	1	1
Nodes	51	112	11
Parameters	112 (crisp)	176 (crisp)	96 (fuzzy triangle numbers)
MSE	0.00013	0.003	0.00004048

Table 3. Comparison performance of different FNN models for dynamic plant identification

In addition, RFNN is more accurate. The simulation using the suggested RFNN demonstrates that the identification error (MSE) is less than the error with other approaches (Table 3).
It can be concluded that DEO learning based RFNN outperform its comparing rivals [25,27] exhibiting considerably lower MSE. In terms of model complexity, the considered RFNN model has lower number of nodes than the models presented in [27].

6.3 Sun-spot prediction

The performance of FRNN was also tested on a well-known problem of sun-spot prediction [8,46]. Sunspot numbers rise and fall with an irregular cycle with a length of approximately 11 years. In addition to this, there are variations over longer periods. The recent trend is upward from 1900 to the 1960s, then somewhat downward. The historical data for this problem were taken from the Internet. Several data sets were prepared as in [8,46]. The data used for training were sun-spot data from years 1700 to 1920. Two unknown prediction sets used for testing were from 1921 to 1955 (PR1) and from 1956 to 1979.
The comparison of performance of the FRNN approach with other existing methods for two different datasets (PR1, PR2) is presented in Table 4 (NMSE i.e. the Normalized Mean Square Error measure is used in these experiments). The last two rows in Table 4 were obtained by two networks trained on the same data sets by two different persons independently (indicated RFNN-1 and RFNN-2, respectively). In RFNN-1 and RFNN2 the total numbers of neurons were 9 (1+7+1) and 13 (1+11+1), respectively. The numbers of connections for RFNN-1 and RFNN-2 were 148 and 179, respectively.
Table 4 presents comparative results on performance of different forecasting methods for sun-spot prediction problem.
As can be seen from Table 4, the suggested RFNN has simpler structure (having only 1 input neuron) than other models. The identification error of the RFNN is less than that of existing models applied to sun-spot forecasting problem.

Author (Method)	Number of inputs	PR1	PR2
Rementeria (AR) [44]	12	0.126	0.36
Tong (TAR) [49]	12	0.099	0.28
Subba Rao (Bilinear) [43]	9	0.079	-
DeGroot (ANN)[15]	4	0.092	-
Nowland (ANN) [36]	12	0.077	-
Rementeria (ANN) [44]	12	0.079	0.34
Waterhouse (HME) [52]	12	0.089	0.27
(RFNN-1)	1	0.066	0.22
(RFNN-2)	1	0.074	0.21

Table 4. MSE obtained by different models for sun-spot prediction

6.4 Application of RFNN to forecast demand for petrol

In this example the problem is to forecast demand for petrol (A92) for optimal scheduling of an oil refinery plant [56]. In our fuzzy forecasting model we assumed the relationship:

$$y(k+1)=F(y(k-2),\, y(k-1)\, ,y(k)) \tag{16}$$

For this example we used actual daily data from existing oil refinery plant for a month period. Approximately 80% of the data (chosen randomly) were used for clustering and training and the remaining data were used for testing of RFNN.

Usually, the structure of RFNN is determined by trial-and-error in advance for the reason that it is difficult to consider the balance between the number of rules and desired performance [20]. In this study, to determine the structure of RFNN, first we convert numeric data into information granules by fuzzy clustering. The number of clusters defines the number of fuzzy rules. By applying the fuzzy C-means clustering method [13,40] on the training data and checking the validity measure suggested in [13] it was identified that an adequate number of clusters is 4. Therefore 4 fuzzy rules were used for the basis for training and further refining. The clustering algorithm identified the following cluster centers for the presented data.

$$\begin{array}{l}
\text{IF } y(t-2) \text{ is A1 AND } y(t-1) \text{ is B1 AND } y(t) \text{ is C1 THEN } y(t+1) \text{ is D1} \\
\text{IF } y(t-2) \text{ is A2 AND } y(t-1) \text{ is B2 AND } y(t) \text{ is C2 THEN } y(t+1) \text{ is D2} \\
\text{IF } y(t-2) \text{ is A3 AND } y(t-1) \text{ is B3 AND } y(t) \text{ is C3 THEN } y(t+1) \text{ is D3} \\
\text{IF } y(t-2) \text{ is A4 AND } y(t-1) \text{ is B4 AND } y(t) \text{ is C4 THEN } y(t+1) \text{ is D4}
\end{array} \tag{17}$$

Initial fuzzy terms A1, A2, A3, A4 were created from the component y(t-2) of the cluster vectors 1, 2, 3, and 4, respectively. Similarly, terms B1, B2, B3, B4 – from y(t-1), C1, C2, C3, C4 – from y(t), and D1, D2, D3, D4 – from y(t+1). The terms A1, A2, ...,B1, B2, ..., C1, C2,...D1, D2, ... are described linguistically.

DEO based training allowed to further decrease MSE of output (forecasting of petrol) after clustering making it ten times lower. The final MSE after training was 0.0008.

6.5 Application of RFNN to control battery charging process

The FRRN designed for battery charging control has 4 inputs, 20 hidden neurons, and 1 output. The four used inputs represent temperature (*T*), change of temperature (*dT*), voltage

(*U*) and change of voltage (*dU*). The output of the controller is the current (*I*) applied for charging the battery.
The network has been trained on the basis of the data base (collected by a separate work group over a year period) contained data series formed of measured temperature, voltage, and current readings from many charging experiments with different batteries.
The proposed control system allows very quick and effective charge of the battery: the charging time is reduced from more than 2000 seconds (with applied constant charge current 2A) to 860 seconds (or even less, if the temperature limit is set higher than 25^{o}C) with dynamically changed (under the control of the proposed intelligent controller) input current. Also the battery is protected from overheating and a long utilization time of the battery can be provided by adequately adjusting the fuzzy rules describing the desired charging process. The results of proposed charging controller compared with other battery chargers for a particular charging experiment (with the same initial conditions) are given in Table 5. The value of decrease in charging time and heating level was $18.0 \pm 2.2\%$ and $5 \pm 0.6\%$, respectively, compared to other methods.

Charging controller	Time (sec)	T_{end}-T_{start}
Proposed approach	860	2,85
FL [4]	(no data)	35-60
FG [14]	959	9
ANFIS [9]	900	50
NeuFuz [16]	1200-1800	5

Table 5. Comparison of different charging controllers

7. Conclusions

In spite of great importance of fuzzy neural networks for solving wide range of real-world problems, unfortunately, little progress has been made in their development.
In this study we have discussed recurrent neural networks with fuzzy weights and biases as adjustable parameters and internal feedback loops, which allows capturing dynamic response of a system without using external feedback through delays. In this case all the nodes are able to process linguistic information.
As the main problem regarding fuzzy and recurrent fuzzy neural networks that limits their application range is the difficulty of proper adjustment of fuzzy weights and biases, we put an emphasize on the RFNN training algorithm.
We have proposed the standard DEO-based method for learning of recurrent fuzzy neural network. The optimization method, customized for RFNN training, compares favorably with the existing gradient-based error minimization method as it is less complex and is more likely to locate the global minimum of network error. As the method does not require derivative information, it is very effective in case when dealing with different distance functions. Also, the considered global optimization algorithm can provide high accuracy of fuzzy mapping with relatively smaller network size.
The RFNN was tested on a number of benchmark identification and time-series forecasting problems well-known in the literature as well as on application problems. Experimental results demonstrated very good performance on all considered problems.

8. References

R.H. Abiyev, "Recurrent Neural Network Based Fuzzy Inference System for Identification and Control of Dynamic Plants", in: Proceedings of International XII Turkish Symposium on Artificial Intelligence and Neural Networks, Vol. 1, No. 1, 2003, 31-39.

D.F. Akhmetov, Y. Dote, S. Ovaska, "Fuzzy Neural Network with General Parameter Adapation for Modelig of Nonlinear Time-Series". IEEE Transactions on Neural Networks, Vol. 12, No. 1, (2001).

F.S.Al-Anzi, A.Allahverdi, "A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times", European Journal of Operational Research, 182 (2007), 80-94.

R.A.Aliev, R.R.Aliev, "Soft Computing and Its Applications", World Scientific, 2001, 465 p.

R.A.Aliev, R.R.Aliev, B.G.Gurimov, K.Uyar, "Dynamic Data Mining Technique for Battery Charging Rules Extraction", Applied Soft Computing Journal (2006), http://dx.doi.org/10.1016/j.asoc.2007.02.015.

R.A.Aliev, R.R.Aliev, B.G.Gurimov, K.Uyar, "Reccurent Neural Network Based System for Battery Charging", Lecture Notes in Computer Science, Springer Berlin, Volume 4492 (2007), 307-316.

R.A. Aliev, B. Fazlollahi, R.R. Aliev, "Soft Computing and Its Applications in Business and Economics", Springer Verlag, 2004, 420 p.

R.A.Aliev, B.Fazlollahi, R.R.Aliev, B.G.Gurimov, "Fuzzy Time Series prediction Method Based on Fuzzy Recurrent Neural Network", Lecture Notes in Computer Science (2006), 860-869.

R.A.Aliev, B.Fazlollahi, R.R.Aliev, B.Gurimov, "Linguistic time series forecasting using fuzzy recurrent neural network", International Soft Computing Journal, Volume 12, Issue 2 (2007), 183-190.

R.A.Aliev, B.Fazlollahi, R.Vahidov, "Genetic Algorithm-Based Learning of Fuzzy Neural Networks. Part 1: Feed-Forward Fuzzy Neural Networks", Fuzzy Sets and Systems 118 (2001), 351-358.

G. Bortolan, "An Architecture of Fuzzy Neural Networks for Linguistic Processing", Fuzzy Sets and Systems, 100 (1998), 197-215.

W.-D. Chang, "Nonlinear system identification and control using a real-coded genetic algorithm", Applied mathematical Modeling 31 (2007), 541-550.

M.-Y. Chen, D.A.Linkens, "Rule-base self-generation and simplification for data-driven fuzzy models", Fuzzy Sets and Systems 142 (2004), 243-265.

O. Cordon, F. Herrera, M. Lozano, "On the Combination of Fuzzy Logic and Evolutionary Computation: a Short Review and Bibliography", in: W.Pedrycz (Ed.), Fuzzy Evolutionary Computation, Kluwer Academic Publishers, Dordrecht, 1997, 33-56.

W.D.DeGroot, "Analysis of Univariate Time Series with Connectionist Nets: A Case Study of Two Classical Examples", Neurocomput., Vol. 3, (1991), 177-192.

D. Dubois, H. Prade (Eds.), "The Handbooks of Fuzzy Sets Series", Kluwer Acad. Publ., 2000.

D. Dubois, H. Prade, R.R. Yager (Eds.), "Fuzzy Information Engineering: A Guided Tour of Applications", Wiley, New York.

Y.Gao, M.J.Er, "NARMAX time series model prediction: Feedforward and recurrent fuzzy neural network approaches", Fuzzy Sets and Systems, Vol. 150, Issue 2 (2005), 331-350.

Y. Hayashi, J. Buckley, E. Czogola, "Fuzzy Neural Networks with Fuzzy Signals and Weights", Internat. J. Intell. Systems 8 (1993), 527-537.

C.-F.Hsu, "Self-organizing adaptive fuzzy neural control for a class of nonlinear systems", IEEE Trans. on Neural Networks, Vol. 18, No. 4 (2007), 1232-1241.

J.-R. Hwang, S.-M. Chan, C.-H. Lee, "Handling Forecasting Problems Using Fuzzy Time Series", Fuzzy Sets and Systems 100 (1998), 217-228.

P.D. Ionescu, M. Moscalu, A. Moscalu, "Intelligent Charger with Fuzzy Logic", in: Proc. of Int. Symp. on Signals, Circuits and Systems, 2003.

H.Ishubuchi, K.Morioka, H.Tanaka, "A Fuzzy Neural Network with Trapezoid Fuzzy Weights", IEEE, New York, 1994, 228-233.

C-F.Juang, "A TSK-Type Recurrent Fuzzy Network for Dynamic Systems Processing by Neural Network and Genetic Algorithms", IEEE Transactions on Fuzzy Systems, Vol. 10, No. 2 (2002), 155-170.

C.-F. Juang, C.-T. Lin. "A Recurrent Self-Organizing Neural Fuzzy Inference Network", IEEE Transactions on Neural Networks, Vol. 10, No. 4 (1999).

C.-F. Juang, K.-C. Ku, "A recurrent network for fuzzy temporal sequence processing and gesture recognition", IEEE Trans. On Systems, Man, and Cybernetics, Part. B: Cybernetics, Vol. 35, Issue 4 (2005), 646-658

C.-H. Lee, C.-C. Teng, "Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks", IEEE Transactions on Fuzzy Systems, Vol. 8, No. 4 (2000), 349-366.

F.H.F. Leung, H.K. Lam, S.H. Ling, P.K.S. Tam, "Tuning of the Structure and Parameters of a Neural Network Using an Improved Genetic Algorithm", IEEE Transactions on Neural Networks, Vol. 14, No. 1 (2003).

C.M. Lin, C.H. Chen, Y.F.Lee, "Recurrent fuzzy neural network adaptive hybrid control for linearized multivariable systems", Journal of Intelligent and Fuzzy Systems, Vol. 17, Number 5 (2006), 479-491.

F.-J.Lin, P.-K.Huang, "Recurrent fuzzy neural network controller design using sliding-mode control for linear synchronous motor drive", IEE Proc.-Control Theory Appl., Vol. 151, No. 4 (2004).

F.-J.Lin, R.-J. Wai, C.-M. Hong, "Hybrid Supervisory Control Using Recurrent Fuzzy Neural Network for Tracking Periodic Inputs", IEEE Transactions on Neural Networks, Vol. 12, No. 1 (2000).

P.Liu, H. Li. "Efficient Learning Algorithms for Three-Layer Regular Feedforward Fuzzy Neural Network", IEEE Transactions on Neural Networks, Vol. 15, No 3, (2004), 545-558.

P.Liu, H.Li, "Fuzzy neural network theory and applications", World Scientific, Singapore (2004), 376 p.

C.-H. Lu, C.-C Tsai, "Generalized predictive control using recurrent fuzzy neural networks for industrial purposes", Journal of Process Control 17 (2007), 83-92

P.Mastorocostas, "A Recurrent Fuzzy Neural Model for Dynamic System Identification", IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics, Vol. 32, No. 2 (2002) 176-190.

S.Nowland., G.Hinton, "Simplifying Neural Networks by Soft Weight-Sharing", Neural Comput., Vol. 4, No. 4 (1992), 473-493.

B.-J.Park, S.-K. Oh, W. Pedrycz, H.-K. Kim, "Design of Evolutionary Optimized Rule-Based Fuzzy Neural Networks Based on Fuzzy Relation and Evolutionary Optimization", in: International Conference on Computational Science (3), 2005, 1100-1103.

W.Pedrycz, "Computational Intelligence. An Introduction", CRC Press, 1997.

W.Pedrycz, “Fuzzy Control and Fuzzy Systems” (second, extended, edition). John Wiley and Sons, New York, 2003.

W.Pedrycs, “Knowledge-Based Clustering: From Data to Information Granules”, John Wiley & Sons (2005), 316 p.

W.Pedrycz, K.Hirota, “Fuzzy vector quantization with particle swarm optimization: A study in fuzzy granulation-degranulation information processing”, Signal Process (2007), doi:10.1016/j.sigpro.2007.02.001.

K.Price, R.Storn, J.Lampinen, “Differential evolution – a practical approach to global optimization, Springer, Berlin (2005).

S.Rao., M.M.Gabr, “An Introduction to Bispectral Analysis and Bilinear Time Series Models”, in Lecture Notes in Statistics. Springer-Verlag, Vol. 24 (1984).

S.Rementeria., X.Olabe, “Predicting Sunspots with a Self-Configuring Neural System”, in Proc. 8th Int. Conf. Information Processing Management Uncertainty Knowledge-Based Systems, 2000.

M. Setnes, H. Roubos, “GA-fuzzy modeling and classi&cation: complexity and performance”, IEEE Trans. Fuzzy Systems 8 (5) (2000) 509–522.

A.Sfetsos, C.Siriopoulos, “Time Series Forecasting with Hybrid Clustering Scheme and Pattern Recognition”, IEEE Trans. on Systems, Man, and Cybernetics – part A: Systems and Humans, Vol. 34, No. 3 (2004), 399-405.

Y.Song, Z.Chen, Z.Yuan, “New chaotic PSO-based neural network predictive control for nonlinear process”, IEEE Trans. on Neural Networks, Vol. 18, No. 2 (2007), 595-600.

S.-F.Su, F.-Y. P. Yang, “On the Dynamic Modeling with Neural Fuzzy Networks”, IEEE Transactions on Neural Networks, Vol. 13, No. 6 (2002).

H.Tong, K.S.Lim, “Threshold Autoregression, Limit Cycle and Cyclical Data”, Int. Rev. Statist. Soc. B, Vol. 42 (1980).

Y.-C. Wang, C.-J. Chien, and C.-C. Teng, “Direct Adaptive Iterative Learning Control of Nonlinear Systems Using an Output-Recurrent Fuzzy Neural Networks”, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 34, No. 3 (2004), 1348-1359.

Y.-J.Wang, J.-S.Zhang, G.-Y. Zhang, “A dynamic clustering based differential evolution algorithm for global optimization”, European Journal of Operational Research 183 (2007), 56-73.

S.R.Waterhouse, A.J.Robinson, “Non-Linear Prediction of Acoustic Vectors Using Hierarchical Mixtures of Experts”, in Advances of Neural Information Processing Systems. Cambridge, MA: MIT Press, Vol. 7 (1995).

S.Wu, M.J.Er, “Dynamic fuzzy neural networks—a novel approach to function approximation”, IEEE Trans. SMC-B 30 (2) (2000) 358–364.

J. Zhang, “Recurrent Neuro-Fuzzy Networks for Nonlinear Process Modeling”, IEEE Transactions on Neural Networks, Vol. 10, No. 2 (1999), 313-325.

Q.-Z. Zhang, W.-S. Gan, Y.-L. Zhou, “Adaptive recurrent fuzzy neural networks for active noise control”, Journal of Sound and Vibration, Vol. 296, Issue 4-5 (2006), 935-948.

Sh.Mehdi, “Fuzzy time series forecasting”, Fourth World Conference on Intelligent Systems for Industrial Automation, 2006, 342-347.