A fast learnt fuzzy neural network for huge scale discrete data function approximation and prediction

Omid Khayat⁎, Javad Razjouyan, Fereidoun Nowshiravan Rahatabad and Hadi Chahkandi Nejad

Abstract. In real world dataset, there are often large amount of discrete data that the concern is the interpolation and/or extrapolation by an approximation tool. Therefore, a training process will be actually used for definition and construction of the approximator parameters. Huge amount of data may lead to high computation time and a time consuming training process. To this concern a fast learnt fuzzy neural network as a robust function approximator and predictor is proposed in this paper. The learning procedure and the structure of the network is described in detail. Simplicity and fast learning process are the main features of the proposing Self-Organizing Fuzzy Neural Network (SOFNN), which automates structure and parameter identification simultaneously based on input-target samples. First, without need of clustering, the initial structure of the network with the specified number of rules is established, and then a training process based on the error of other training samples is applied to obtain a more precision model. After the network structure is identified, an optimization process based on the known error criteria is performed to optimize the obtained parameter set of the premise parts and the consequent parts. At the end, comprehensive comparisons are made with other approaches to demonstrate that the proposed algorithm is superior in term of compact structure, convergence speed, memory usage and learning efficiency.

Keywords: Self-organizing fuzzy neural network, hybrid learning algorithm, function approximation, prediction, chaotic time series

1. Introduction

Neuro-fuzzy systems, as hybrid systems, have become popular in the last decade [1, 3, 7–9, 11, 18, 20, 21, 25, 27–31, 33–36, 38]. They are combinations of the theories of fuzzy logic and neural networks. Usually, fuzzy neural networks (FNNs) are hybrid systems in which the fuzzy techniques are actually used to create or enhance neural networks [21]. However, some researchers do not distinguish between neuro-fuzzy and fuzzy-neural systems and interchangeably use both the terms to describe the same hybrid system. Fuzzy systems, as model-free approaches, provide a high-level, approximate human reasoning ability. It has been shown that fuzzy systems could serve as a powerful tool for system modeling and control [9, 36]. A fuzzy system is intrinsically a rule-based system which is composed of a set of linguistic rules in the form of “IF-THEN”. As there is no formal and effective way of knowledge acquisition, it is difficult for a designer to

⁎Corresponding author. Omid Khayat, Young Researchers Club, South Tehran Branch, Islamic Azad University, Tehran, Iran. E-mails: khayat@aut.ac.ir, okh2314@yahoo.com (Omid Khayat), razjouyan@aut.ac.ir (Javad Razjouyan), nooshiravan@gmail.com (Fereidoun Nowshiravan Rahatabad), Hadi64.msc@gmail.com (Hadi Chahkandi Nejad).
acquire appropriate fuzzy rules from numerical training data [39]. In addition, fuzzy systems lack adaptability for possible changes in the reasoning environment.

Fuzzy neural networks (FNNs) are hybrid systems that combine the theories of fuzzy logic and neural networks, thus can make effective use of easy interpretability of fuzzy logic, as well as superior learning ability and adaptive capability of neural networks. Such integration overcomes the drawbacks of fuzzy systems mentioned above and renders neuro-fuzzy systems more powerful than either one alone. FNNs are widely used in areas of adaptive control, adaptive signal processing, nonlinear system identification, pattern recognition, and so on [10, 11, 20, 38].

The design of FNNs consists of structure and parameter identification. Parameter identification involves determining parameters of premises and consequences. Structure identification comprises the partitioning of input–output space and determination of the rule number for the desired performance.

Various means adopted for the partitioning of input–output space can be divided into two broad categories [35]: (1) Static adaptation method, where the number of input–output partitions is fixed, while the corresponding fuzzy rule configurations are adapted by different optimization algorithms. One of the early famous works is the adaptive-network-based fuzzy inference system (ANFIS) [8]. (2) Dynamic adaptation method, where both the number of input–output partitions and their corresponding fuzzy rule configurations are adapted simultaneously [1, 28].

In general, the partition of input–output space is grid type, whose number depends on the number of input variables and the number of membership functions for each input variable [8]. Such a measure suffers mainly from two drawbacks: (1) The number of input–output space partitioning needs to be given by prior expert knowledge and keeps fixed during learning, which is prone to bring a deficiency or redundancy of fuzzy rules. (2) The grid-type partitioning, when used for complex systems, suffers from the curse of dimensionality, as the number of inputs becomes larger [18].

This method has been introduced and presented partially earlier by the Authors in CSICC 2009 conference [12] concerning mainly on the high convergence speed and low memory usage of training process with focus on the approximation and prediction capability. In this paper, the method is intentionally described again aiming both for availability of the paper more for readers in an International Journal and also accenting on the time analysis of the method and the structure performance as a fast learnt SOFNN. The recursive and sample-based proposed hybrid algorithm is described as follows. In each training epoch, based on the criteria error and the error of the training samples, one sample is selected in order to training the network. A Training Validity Index (TV) is defined to find the best sample for training the network. Without need of clustering the input or output space and without a priori knowledge about the relationship between the input and output space, an initial structure with the minimum fuzzy sets is first constructed. An optimization process is then applied to network to modify the parameters gaining better results and more precise model. If the conditions are not satisfied, the sample with maximum TV is chosen to train the network aiming a more precise model.

This paper is organized as follows: Section 2 describes the architecture of the proposed FNN. In Section 3, the proposed algorithm including the coarse tuning phase and the optimization phase is introduced. To demonstrate the effectiveness of the proposed approach, Section 5 presents several applications in different areas, where comprehensive comparisons are made with other learning algorithms. Finally conclusions are drawn in Section 6.

2. Architecture of the proposed fuzzy neural network

The architecture of the proposed FFPNN is a five layer FNN shown in Fig. 1 similar to [26]. Without loss
of generality, we consider a multi-input single-output (MISO) fuzzy model which consists of L rules:

\[ R_k: \text{IF } x_1 \text{ is } A_{1k}^1 \text{ AND } \ldots \text{ AND } x_n \text{ is } A_{nk}^n \text{ THEN } y_k \text{ is } w_k. \]

Where \( R_k \) is the \( k \)-th rule \((1 \leq k \leq L)\), \( x = [x_1, \ldots, n] \) is input linguistic vector, \( w_k \) is the \( k \)-th consequent parameter. \( A_{ik}^i \) are fuzzy sets defined by the membership functions.

If the membership functions are Gaussian functions:

\[ \mu_{ik}(x_i) = \exp \left\{ -\frac{(x_i - c_{ik})^2}{\sigma_{ik}^2} \right\} \]

\( i = 1, 2, \ldots, n \) \& \( k = 1, 2, \ldots, L \)  

(1)

Where \( \mu_{ik} \) is the \( k \)-th membership function of variable \( x_i \), \( c_{ik} \) and \( \sigma_{ik} \) are the center and sigma of the Gaussian function, respectively, \( n \) is the number of input variables and \( L \) is the number of membership functions for each input variable.

If the membership functions are Triangular functions:

\[ \mu_{ik}(x_i) = \begin{cases} 1 & \text{if } x_i \leq \text{left}_{ik} \\ \frac{x_i - \text{left}_{ik}}{\text{d}_1} & \text{if } \text{left}_{ik} < x_i \leq \text{center}_{ik} \\ \frac{\text{right}_{ik} - x_i}{\text{d}_2} & \text{if } \text{center}_{ik} < x_i \leq \text{right}_{ik} \\ 0 & \text{otherwise} \end{cases} \]

\( i = 1, 2, \ldots, n \) \& \( k = 1, 2, \ldots, L \)  

(2)

Where \( \mu_{ik} \) is the \( k \)-th membership function of variable \( x_i \), \( \text{center}_{ik} \) is the center of Triangular function, and \( \text{left}_{ik} \) and \( \text{right}_{ik} \) are the left width and right width of Triangular function respectively, \( n \) is the number of input variables and \( L \) is the number of membership functions for each input variable.

Layer 1: Input layer. Each node in layer 1 represents an input linguistic variable.

Layer 2: Fuzzification layer. Nodes in layer 2 are arranged into \( L \) groups, each group representing the IF-part of a fuzzy rule. Each node computes the value of membership function \( \mu_{ik}(x_i) \).

Layer 3: Fuzzy Inference layer. Product inference method is used here; the number of nodes in layer 3 is equal to the number of fuzzy rules. Each node computes the rule activation strengths, for the \( k \)-th rule \( R_k \), its output is:

\[ \mu_k(x) = \prod_{i=1}^{n} \mu_{ik}(x_i), \quad k = 1, 2, \ldots, L. \]  

(3)

Layer 4: Normalized layer. The number of \( N \) (normalized) nodes is equal to that of \( R \) nodes. The output of \( \phi_k(x) \) is:

\[ \phi_k(x) = \frac{\mu_k(x)}{\sum_{k=1}^{L} \mu_k(x)}, \quad k = 1, 2, \ldots, L. \]  

(4)

As to conventional FNNs, the number of membership functions of each input variable needs to be given according to expert knowledge. However, different fuzzy systems may differ from one another significantly, it is difficult for a designer, and even he/she is a domain expert, to find the appropriate number of rules for the specific systems. Moreover, the structure of FNNs keeping fixed results in the lack of adaptability for different objects. A deficiency or redundancy of fuzzy rules is unavoidable. This leads us to develop a new learning algorithm which is capable of automatically determining the number of fuzzy rules for the desired system performance.

3. The novel hybrid learning algorithm

This revisited algorithm is designed primarily for use in the offline situations. It includes two stages. In the first stage, the initial structure is generated based on a set of neuron adding method discussed later. The optimization process is therefore used in the second stage and attempts to modify the parameters to obtain a more precision model. Without lose of generality, first it is assumed that the system is single-input single-output (SISO).

3.1. The coarse tuning phase

An initial structure of the FNN is first constructed with minimum of fuzzy sets, after that, nodes and links are added to form the final structure of the network as the learning proceeds.

The availability of a dataset composed by \( N \) input–output pairs is assumed

\[ S = \{\{x_j, y_j\} : j = 1, 2, \ldots, N\} \]  

(5)

Where \( x_j \) and \( y_j \) are the inputs and outputs, and \( N \) is the number of training samples. In each epoch of
training, one sample is selected in order to add a node and its links. The selected sample should be the most appropriate one. For this sake, Training Validity Index is defined to distinguish the best one. Assume that the input space is a \( P \)-dimensional space and the error is as follow

\[
E = \{E_{ij}\}_{i=1,2,...,P, j=1,2,...,N}
\]

(6)

In which, \( E_{ij} \) is the error of network in the place of sample \( x_j \). For the dimension of \( p_{ij} = 1,2,...,P \), \( C_{p} \) is defined as the ordered set of \( E_{p}(j=1,2,...,N) \) which are zero

\[
C_{p} = \{C_{j}\}_{j=1,2,...,N}
\]

(7)

In which, \( C_{(1,2,...,p)} \) is where in the dimension \( p \) that the error is zero and \( R \) is the number of zero in the error set. The Training Validity for sample \( x_j \) is defined as below

\[
TV_j = \prod_{p=1}^{P} TV_{pj}
\]

(8)

\[
E_{pj} = |\hat{y}_{pj} - y_{pj}|
\]

(10)

Where \( p \) is the dimension, \( C_{(p+1)} \) and \( C_p \) are sequent members of set \( C_p \) subject to \( C_p \leq \hat{y}_{pj} \leq C_{(p+1)} \) and \( y_{pj} \) and \( \hat{y}_{pj} \) are the desired and calculated output, respectively. Training Validity of each sample is calculated using the Equations (8)-(10). As it can be seen from Equation (9), \( TV \) calculation depends on the criteria error. The criteria error may be Mean Squared Error (MSE), Average Percentage Error (APE) or maximum error of samples.

3.2. Optimization process

After the network structure is established, the network enters the optimization-learning phase. Initialize parameters of the network with the values gained in the learning phase. An improved back propagation (BP) algorithm [3, 29, 30] with adaptive learning rate and momentum is adopted for optimizing the parameters based on the same dataset \( S \); thereby the final network model is obtained.

According to the Equations (1–5), for an arbitrary input \( x_p \) from \( S \), the output of the network is:

\[
y_p = \frac{\sum_{k=1}^{L} \mu_k(x_p)w_{k1}}{\sum_{k=1}^{L} \mu_k(x_p)}
\]

(11)

Assume the error function is MSE:

\[
E = \sum_{p=1}^{P} |y_p - \hat{y}_p|^2
\]

(12)

Or APE:

\[
E = APE = \frac{1}{P} \sum_{p=1}^{P} \frac{|y_p - \hat{y}_p|}{|y_p|} \times 100\%.
\]

(12)

And or Maximum error of samples as:

\[
E = \text{Max}(|y_p - \hat{y}_p|)_{p=1,2,...,P}.
\]

(12)

Where \( y_p \) and \( \hat{y}_p \) are the \( p \)th calculated output and desired output, respectively. If the error function is assumed as (12):

\[
\frac{\partial E}{\partial \theta_{jk}} = \frac{\partial E}{\partial y_p} \frac{\partial y_p}{\partial \theta_{jk}}
\]

(13)

\[
= \left( y_p - \hat{y}_p \right) \left( \frac{w_k - y_p}{\sum_{k=1}^{L} \mu_k(x_p)} \right)
\]

\[
\mu_k(x_p) \frac{\delta \sigma_{c_k}^2}{\sigma_k^2}
\]

The following Equation (14) can be obtained from the Equation (13), and the results of Equations (15), (16) can be similarly gained:

\[
\Delta \theta_{jk} = -\eta \sum_{p=1}^{P} \frac{\partial E}{\partial \theta_{jk}}
\]

(14)
\[ \Delta \sigma_{ik} = -\eta \sum_{p=1}^{P} \frac{\partial E}{\partial \sigma_{ik}} = -\eta \sum_{p=1}^{P} (y_p - \hat{y}_p) \left( \frac{w_k - y_p}{\sum_{k=1}^{L} \mu_k(x_p)} \right) \times \mu_k(x_p) \frac{2(x_p - c_{ik})^2}{\sigma_{ik}^3} \]  

\[ \Delta w_{ik} = -\eta \sum_{p=1}^{P} \frac{\partial E}{\partial c_{ik}} = -\eta \sum_{p=1}^{P} (y_p - \hat{y}_p) \left( \frac{\mu_k(x_p)}{\sum_{k=1}^{L} \mu_k(x_p)} \right) \]  

\[ i = 1, \ldots n \quad p = 1, \ldots P \quad k = 1, \ldots L, \]  

where \( c_{ik}, \sigma_{ik} \) and \( w_{ik} \) are updated according to Equations (14)–(16), respectively. We use above updating formulas also for (\( \dot{12} \)) and (\( \ddot{12} \)).

### 3.3. Algorithm Description

The algorithm we propose is based on an iterative find and train till the termination criterion is reached. After the initialization step, the algorithm continues by computing the Training Validity Index (TV) of the training samples. A simple search method is then applied to find the sample with maximum TV which is considered to be the best sample to be used for training in the next epoch. The proposed hybrid algorithm is summarized as below:

1) An initial structure with the minimum fuzzy rules (predefined) is constructed.
2) Termination criteria? Yes: STOP, No: continue.
3) Calculate the set TV = \{TV\} = 1, 2, ..., N
4) Determine the Error criterion for training procedure.
5) Train the network by the sample with maximum TV index.
6) Adjust the parameters using optimization process.
7) Go to step 2.

### 4. Simulation Results

In this section, the effectiveness of the proposed algorithm is demonstrated in the examples of function approximation and prediction. All the programs were developed using MATLAB 6.5 software and each problem was simulated on a Pentium IV 1.8 GHz desktop computer.

#### 4.1. Static function approximation

For the sake of comparison, one example is tested to verify the validity of the proposed algorithm. The function is a nonlinear function (F1) taken from [13] and expressed as:

\[ F1 = (1 + x_1^2 + x_2^{1.5})^2, \quad 1 \leq x_1, x_2 \leq 5 \]

Fifty input output data are used to build the fuzzy model and another 100 data are used for evaluating the performance [32]. The MSE curve against the number of rules with 200 epochs is shown in Fig. 2. The obtained MSE for 3 rules is 0.0021. Table 1 depicts the comparison results for function F1.

<table>
<thead>
<tr>
<th>Function algorithms</th>
<th>Number of rules</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sugeno and Yasukawa</td>
<td>6</td>
<td>0.079</td>
</tr>
<tr>
<td>Nozaki et al.</td>
<td>25</td>
<td>0.0085</td>
</tr>
<tr>
<td>FCRM</td>
<td>3</td>
<td>0.009</td>
</tr>
<tr>
<td>Kim et al. (1997–1998)</td>
<td>3</td>
<td>0.0197</td>
</tr>
<tr>
<td>Tsekouras et al.</td>
<td>6</td>
<td>0.0148</td>
</tr>
<tr>
<td>The proposed algorithm without optimization</td>
<td>3</td>
<td>0.032</td>
</tr>
<tr>
<td>The proposed algorithm</td>
<td>3</td>
<td>0.0021</td>
</tr>
</tbody>
</table>

![Fig. 2. The MSE for function F1.](image-url)
the result of the proposed algorithm with Gaussian membership function along with the results obtained by other approaches that can be found in the literature. It is shown that the optimization phase indeed performs better performance than that of the proposed algorithm without it [14] utilizes a gradient descent method based fuzzy model for function approximation. The comparative results show that the proposed algorithm performed better than PCRSM and [14] model. Nozaki et al. uses a considerably large number fuzzy rules to reduce the value of MSE, but the propose algorithm uses less fuzzy rules and obtains better performance than [23]'s model. In Table 1, it is shown that the proposed algorithm indeed outperforms other approaches.

4.2. Chaotic time series: Mackey–glass time series

In this simulation, the Prediction of Chaotic Time Series is used to compare the convergence speed of the proposed algorithm with some other learning algorithms. The Mackey Glass chaotic time series [2, 8, 21, 22, 24] in consideration here is generated from the following delay differential equation

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

Crowder [5] extracted 1000 input–output data pairs \{x, y\} which consist of four past values of \(x(t)\), i.e.

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

Where \(\tau = 17\) and \(x(0) = 1.2\). There are four inputs to model, corresponding to these values of \(x(t)\) and one output representing the value \(x(t + \Delta t)\), where \(\Delta t\) is a time prediction to the future. The first 500 pairs (form \(x(1)\) to \(x(500)\)) are the training data set, while the remaining 500 pairs (form \(x(501)\) to \(x(1000)\)) are the testing data set used for validating the proposed algorithm.

Cheng-Jian Lin and Shang-Jin Hong (2007) achieved the RMSE equally to 0.000243 after 500 generations. For comparison with the results of Cheng-Jian’s algorithm, we do these simulations in the same conditions and with the same parameters. The number of membership functions and rules are determined based on continuing the training process until the required RMSE is achieved. After 200 training epochs, the RMSE is converged to 0.0002 for the training phase. The learning curve is shown in Fig. 3(a). As it can be seen from the Table 2, for the same RMSE the convergence speed of the proposed algorithm is faster than that of the algorithm of Cheng-Jian Lin and Shang-Jin Hong (2007) and Lin and.
Table 3

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of inputs</th>
<th>Number of rules</th>
<th>Number of parameters</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANFIS [15]</td>
<td>4</td>
<td>16</td>
<td>104</td>
<td>0.0018</td>
</tr>
<tr>
<td>Chen et al. [2]</td>
<td>–</td>
<td>13</td>
<td>–</td>
<td>0.0163</td>
</tr>
<tr>
<td>Cho and Wang [3]</td>
<td>–</td>
<td>13</td>
<td>–</td>
<td>0.0131</td>
</tr>
<tr>
<td>Nauck and Kruse [22]</td>
<td>–</td>
<td>26</td>
<td>–</td>
<td>0.0154</td>
</tr>
<tr>
<td>Paul and Kumar [24]</td>
<td>–</td>
<td>9</td>
<td>–</td>
<td>0.0055</td>
</tr>
<tr>
<td>Kakdal [10]</td>
<td>4</td>
<td>9</td>
<td>117</td>
<td>0.0001</td>
</tr>
<tr>
<td>De Souza et al. [6]</td>
<td>4</td>
<td>–</td>
<td>64</td>
<td>0.0065</td>
</tr>
<tr>
<td>Wang and Mendel [37]</td>
<td>9</td>
<td>121</td>
<td>–</td>
<td>0.01</td>
</tr>
<tr>
<td>Kim [13]</td>
<td>4</td>
<td>9</td>
<td>81</td>
<td>0.0264</td>
</tr>
<tr>
<td>Tsekouras et al. [32]</td>
<td>4</td>
<td>6</td>
<td>78</td>
<td>0.0041</td>
</tr>
<tr>
<td>The proposed algorithm</td>
<td>4</td>
<td>9</td>
<td>75</td>
<td>0.0011</td>
</tr>
</tbody>
</table>

Table 4

<table>
<thead>
<tr>
<th>Model</th>
<th>Inputs</th>
<th>Rules</th>
<th>Training time (msec)</th>
<th>Relative evaluation time</th>
<th>Desired RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANFIS (MATLAB)</td>
<td>4</td>
<td>16</td>
<td>Gaussian Mem. Func.</td>
<td>~3300</td>
<td>1.4</td>
</tr>
<tr>
<td>ANFIS (MATLAB)</td>
<td>4</td>
<td>16</td>
<td>Triangular Mem. Func.</td>
<td>~4700</td>
<td>1.4</td>
</tr>
<tr>
<td>POLYNOMIAL</td>
<td>4</td>
<td>–</td>
<td>Order: (10, 10)</td>
<td>~2200</td>
<td>0.92</td>
</tr>
<tr>
<td>POLYNOMIAL</td>
<td>4</td>
<td>–</td>
<td>Order: (50, 50)</td>
<td>~8000</td>
<td>0.96</td>
</tr>
<tr>
<td>Feedforward NN</td>
<td>4</td>
<td>–</td>
<td>(1, 14), Triangular, 500</td>
<td>~8000</td>
<td>1</td>
</tr>
<tr>
<td>Case 1</td>
<td>4</td>
<td>9</td>
<td>Gaussian Mem. Func.</td>
<td>~800</td>
<td>1</td>
</tr>
<tr>
<td>Case 2</td>
<td>4</td>
<td>9</td>
<td>Triangular MSE criterion</td>
<td>~450</td>
<td>1</td>
</tr>
<tr>
<td>Case 3</td>
<td>4</td>
<td>9</td>
<td>Triangular APE criterion</td>
<td>~550</td>
<td>1</td>
</tr>
<tr>
<td>Case 4</td>
<td>4</td>
<td>9</td>
<td>Triangular Max. Error criterion</td>
<td>~750</td>
<td>1</td>
</tr>
</tbody>
</table>

The result of prediction is given in Fig. 3(b). Table 3 compares the performance of the resulted model to the performances of other methods that can be found in the literature, in terms of RMSE and the number of rules. This table indicates that the model produced by the proposed algorithm is superior to all other models, except from the Tsekouras et al. [32] in the term of number of rules and De Souza et al. [6] in the term of number of parameters.

4.3. Computation time analysis

In this section, the proposed algorithm is analyzed in terms of the minimum time required for the training process and also the computation time of the structure for an arbitrary set of input points. For evaluation of an approximation tool the approximation error may be the most important factor. Sometimes, hug amount of data are given for the interpolation and extrapolation purpose. For example, an accurate prediction may be requested with an available dataset in thousands of periods in an economic problem. With short periods of time and needs of an online prediction, the speed of response may be as important as the accuracy. Then, a compromise should be made between the accuracy and the time of prediction process. Mackey-Glass Time series is considered again for time analysis of the structure and the training process. The proposed method in 4 schemes is compared with 2 same structures of ANFIS (MATLAB Toolbox) one with Gaussian membership function and another with Triangular membership function, 2 polynomials one with orders in (10, 10) range and another with order in (50, 50) range and a Feed Forward Neural Network. ANFIS as a neuro-fuzzy system has high capability in function approximation and prediction but slightly time consuming. Polynomials are also efficient in function modeling but it needs high order range to achieve the desired RMSE of 0.001. This leads to a little high computation time since the optimum order should be found first in the predetermined range.

Table 4 gives the results of the comparison in computation time of training and evaluation process for the proposed algorithm and structure. As discussed before, for a desired RMSE the proposed structure has the lowest computation time among the methods given in the table. This comparison is made for both Triangular and Gaussian membership functions and also for 3 types of criteria. In all cases of training procedure and most of evaluation processes the proposed algorithm yields better performance and time characteristics.
5. Discussions and conclusion

For the cases where a large amount of discrete data are existed and the concerns are the interpolation and/or extrapolation by an approximation tool a learnt fuzzy neural network as a robust function approximator and predictor is proposed in this paper. The learning procedure and the structure of the network were described in detail. Simplicity and fast learning process are the main features of the proposing Self-Organizing Fuzzy Neural Network (SOFNN), which automates structure and parameter identification simultaneously based on input-target samples.

Simulations show that the FPFPNN has the ability to obtain a more compact structure with higher accuracy than those of previous works, though it can realistically be used in the offline situations only and periodically for online applications. Additionally, in task of computation time and memory usage the superiority of the proposed algorithm is depicted by the results of experiments. Learning process is the core of time consuming procedures in the SOFNN applications. By the proposed fast learnt algorithm compared to ANFIS structure, Polynomial and Feed Forward Neural Network periodic online applications are more convenient and justified. The reasons why the proposed structure perform approximation and prediction fast include but not limited to direct structure and parameter identification based on available dataset, using Training Validity Index for directing learning procedure, low parameter sets, lower fuzzy rules therefore lower parameter computation time and memory usage the superiority of the proposed algorithm is depicted by the results of experiments.

input-target samples.

References


