An Approach for Generation of Perception Frames based Fuzzy Neural Network from Data

ALEKSANDAR MILOSAVLJEVIĆ, LEONID STOIMENOV, DEJAN RANČIĆ
Computer Science Department, Faculty of Electronic Engineering
University of Niš
Aleksandra Medvedeva 14, 18000 Niš
SERBIA
alexm@elfak.ni.ac.yu http://gislab.elfak.ni.ac.yu/alexm

Abstract: - In this paper we present an approach for generation and initialization of fuzzy neural networks (FNN) from data. Fuzzy neural networks are concept that integrates some features of the fuzzy logic and the artificial neural networks theory. Based on analysis of several different fuzzy neural networks models, uniform representation method is presented, and two basic types are identified: FNN based on perception frames, and FNN with independent rules. Presented algorithm supports generation of fuzzy neural network based on perception frames through following steps: clustering of a training set, identification of input variables perception frames, generation of rules using training set, and training for adaptation using gradient descent method.

Key-Words: - Fuzzy neural networks (FNN), Rule generation, Perception frames, Generalized neurons

1 Introduction

Fuzzy neural networks (FNN) are based on the integration of fuzzy logic and artificial neural networks – theories that already have taken their places as major interests for researchers in the field of the artificial intelligence.

Fuzzy logic, based on the Zadeh's fuzzy sets, has mathematical potential for describing indeterminacy related with human cognitive processes, such as thinking and reasoning. Fuzzy logic enables reasoning based on incomplete and imprecise information, known as approximate reasoning. On the other hand, artificial neural networks, with their diverse architectures built on the concept of an artificial neuron, are developed to ape biological neural systems in performing functions such as learning or pattern recognition. While fuzzy logic enables mechanism for reasoning based on incomplete and imprecise information, artificial neural networks provide some remarkable abilities such as learning, adaptation and generalization.

Neural networks so as fuzzy logic are dealing with important aspects of knowledge representation, reasoning and learning, but in different approaches with their advantages and weaknesses. Neural networks can learn from the examples, but it is nearly impossible to describe knowledge acquired in that way. On the other hand, fuzzy logic, that enables approximate reasoning, does not have ability of self-adaptation. Using these complementary theories, fuzzy neural networks (FNN) found their greatest application in implementation of the adaptive fuzzy controllers.

The paper is structured as follows: in the second section, theoretical bases and classification of some existing solutions for FNNs realization are presented. This section also describes different training mechanism application levels and some of existing training algorithms described in the literature. The third section of this paper describes proposed algorithm for automatic generation of FNNs based on perception frames. The fourth section represents evaluation of proposed algorithm showing some test results. Finally, in the conclusion, the achieved results are summarized.

2 Fuzzy Neural Networks with Fuzzy Singleton Reasoning

Implementation of FNNs requires certain extension of artificial neuron functions. These extensions should provide model of artificial neuron efficient to realize fuzzy logic operations [1]. Analyzing mathematical model of neuron known as *perceptron* [2] it is possible to describe it as process element that involves three successive operations. *Synaptic* operation describe interaction of an input with the corresponding weight, *aggregation* operation aggregates outputs from synapses generating an input for the third and the last *activation* operation.

Perceptron has product as synaptic operation, addition as aggregation, while the activation operation is defined by the agreed activation function. Generalization of these operations leads to concept known as ***generalized neuron*** [1]. Process elements attained in this way by their functions have nothing in common with natural neurons like the perceptron does, but the whole theory of neural networks, and especially training, can be successfully applied. To specify type of generalized neuron three letters are used. Letters defines agreed operations, so for example mark "PSS" describes generalized neuron with product operation on the synapses, sum (addition) as the aggregation and sigmoidal function as the activation; a neuron specified in that way corresponds to perceptron.

It is already mentioned that FNN can act as adaptive fuzzy controllers. Fuzzy controllers, described in [3], contain *fuzzyfication block* that converts input signals from external (non-fuzzy) state into internal fuzzy state, *reasoning block* that according to premises membership degree applies given rules, and finally *defuzzyfication block* that converts internal fuzzy conclusion into external (non-fuzzy) outputs suitable for control. Using networks of generalized neurons with appropriately selected operations it is possible to realize each of these blocks. Solutions are not unique and some of them are suggested in [1] and [4]. But, this approach for realization of adaptive fuzzy controllers often does not give the most efficient results by the complexity. Because of that, in practice it is the most often used simplified model of fuzzy reasoning – ***fuzzy singleton reasoning***. Definition of the fuzzy singleton reasoning, so as derivation of needed expressions are given in [1], [5], and [6]. Different solutions for realization of singleton defuzzyfier are suggested in [1], [7] and [8].

There are two basic configurations of FNNs based on the fuzzy singleton reasoning: (1) *FNNs based on perception frames*; (2) *FNNs with independent rules*. The first configuration is adapted from [8], [9], [10], and [11]; and the second from [7] and [8]. The fig. 1 and 2 represent these architectures on the examples of networks with 2 inputs and 1 output parameter.

2.1 FNN based on Perception Frames

FNN based on perception frames (see fig. 1) can be realized with four layers of generalized neurons. Marks of the neurons per layers are: III, QIE, IPI and PCI. The task of the first layer neurons is to transmit input signals to neurons in the next layer; letters "I" signifies identity function ($f(x)=x$). The QIE neurons from the second layer determine membership degree of the inputs with the corresponding partitions from the parameters perception frames. Fuzzy sets of the partitions that forms a perception frame are defined with complex weights ($w=(a,b)$) of these neurons. The first component defines center, and the second width of the fuzzy set represented by the Gaussian-type function. Letter "Q" signifies synaptic operation square distance calculated as $f(x,w)=(x-a)^2/b^2$, while letter "E" signifies exponential activation function $f(x)=exp(-x)$. Each neuron from the IPI layer represents a rule and gives agreement degree with the premises as the output. Letter "P" in this case signifies that product is used for the aggregation of the corresponding inputs. Finally, the PCI neuron in the last layer acts as a singleton defuzzyfier that uses center of mass defuzzyfication method. Weights of this neuron represent fuzzy singleton consequences. The center of mass method is implemented with PCI neuron that has products operation on the synapses, while aggregation is done by dividing a sum of the synapse outputs with the sum of the inputs ($y = \Sigma wx / \Sigma x$).

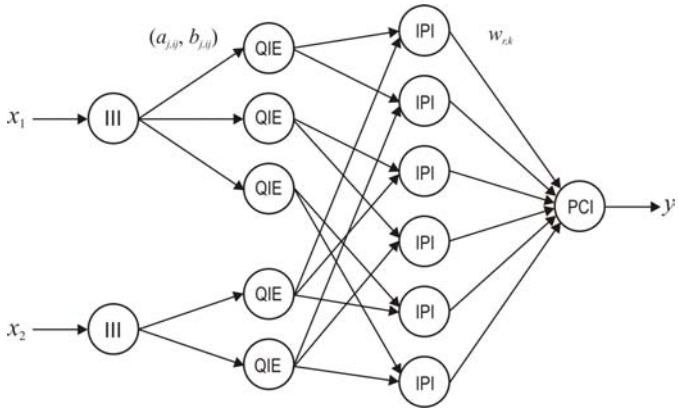


Fig. 1. FNN based on perception frames

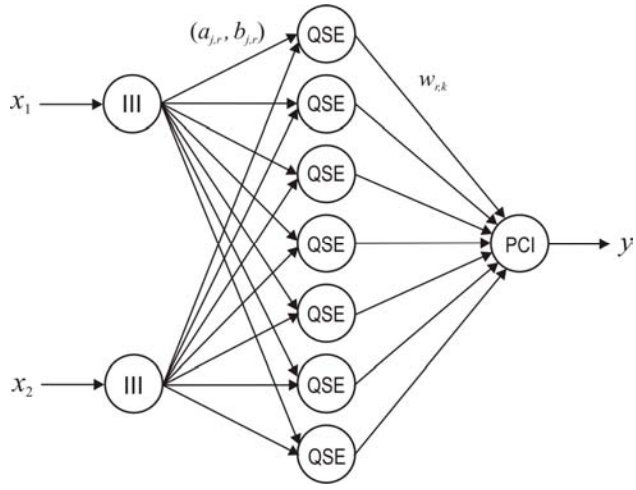


Fig. 2. FNN with independent rules

2.2 FNN with Independent Rules

FNN with independent rules (see fig. 2) is realized as network with only three layers of neurons. Beside III and PCI neurons described above, we have here new type of generalized neurons marked as QSE. Each of these neurons represents a rule. These rules are called independent because they are not associated with the partitions of the perception frames. On the contrary, all parameters needed for a rule definition are contained in the complex weights of the neuron. In contrast of QIE neurons that had only one input, QSE neurons have as much inputs as input parameters of the network. Letter "S" in the mark signifies that sum is used for aggregation of the synapse outputs.

Principal difference of these two models is shown on fig. 3. Fig. 3.a represents example of fuzzy rules distribution over an input domain of some FNN based on perception frames, while fig. 3.b represents distribution for FNN with independent rules. Both FNNs have 2 input parameters x_1 and x_2. Fig. 3.a additionally shows fuzzy partitions (fuzzy sets) of each input parameter. The rules are formed as combinations of these partitions, i.e. locked with in partitions of perception frames. On the contrary, each premise of a rule in the FNN with independent rules is defined by its own parameters, so, as it can be seen, distribution is much more flexible. Of course, price of flexibility is lack of ability to easily describe knowledge gathered by the FNN using fuzzy-linguistic rules.

2.3 Fuzzy Neural Network Training

Adaptability of a fuzzy controller realized using generalized neurons is obtained by applying neural network training mechanisms. Problem of training FNNs can be observed in four different levels [1]. In case of adaptive fuzzy controllers, the most important is *training for adaptation* and *training for rules generation.*

Training for adaptation demands predefined fuzzy rules and membership functions, and the mechanism of neural networks supervised learning is applied only to adjust parameters that describes these fuzzy membership functions. For that purpose extended *backpropagation* algorithm is used. The extensions of the algorithm make it applicable to neurons with different synaptic, aggregation and activation operations. Basic backpropagation algorithm is described in [2], while all corresponding modifications for FNNs with independent rules are described in [8], and for FNNs based on perception frames in [10].

Training for rule generation is the most complex level of learning where the whole fuzzy system is automatically generated from the data that describes it. This type of training is usually organized into two phases. In the first phase numerical inputs and outputs are grouped into clusters. Later, each cluster is being transformed into fuzzy rule. The second phase takes the initially generated network and applies training for adaptation to increase the system performance. Clustering algorithm, described in [7], that uses the nearest neighbor method is applicable only to FNNs with independent rules.

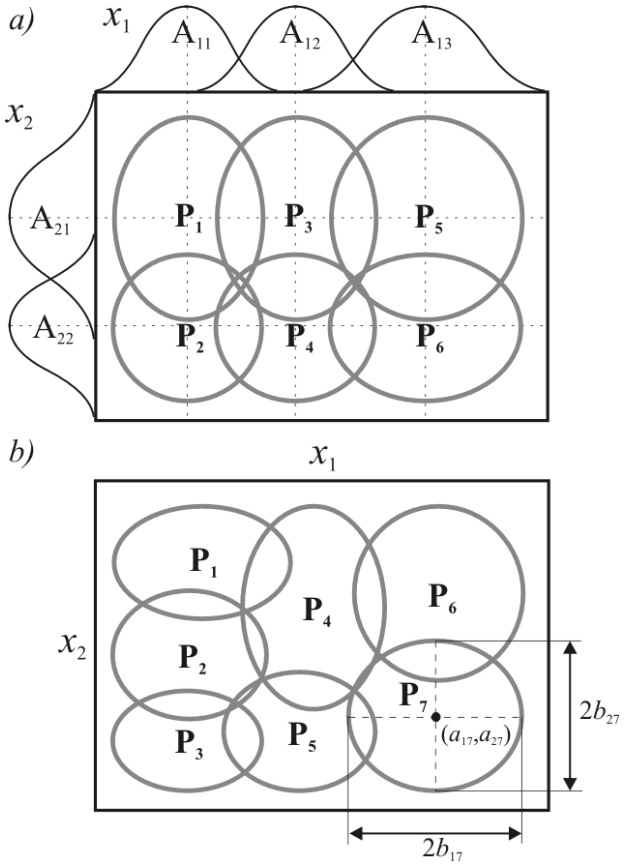


Fig. 3. Fuzzy rules distribution over an input domain for FNN based on perception frames (a), and FNN with independent rules (b).

3 Automatic Generation of FNN based on Perception Frames

The algorithm that we are presenting supports rule generation of fuzzy neural network based on perception frames, while training for adaptation relies on gradient decent method, i.e. modified backpropagation algorithm.

The suggested algorithm is organized in following steps:

1. Clustering of an input training set.

2. Determining fuzzy set number and centers in perception frames of input parameters.
3. Setting initial widths of perception frames fuzzy sets.
4. Generation of rules using training set.
5. Training for adaptation using gradient descent method.

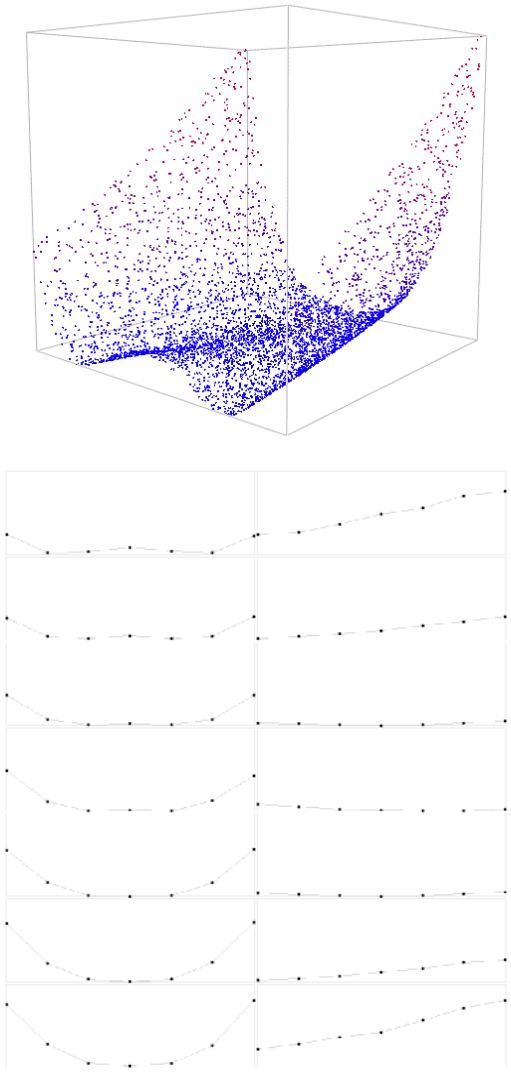


Fig. 4. Partial output functions extracted from a cluster set

3.1 Clustering of an input training set

Purpose of this phase is to simplify a training set, and to filter the data of an eventual noise. But also, substitution of a training set with clusters is demand of the next phase.

An input for this phase of the algorithm is:

a) a training set consisted of inputs and an expected output pairs,
b) number of partitions of an each input parameter, and
c) number of partitions of an output.

A training set is represented as an array of vectors where each vector is one training pair consisted of inputs and corresponding output value. Inputs and an output values should be normalized to an interval [0, 1]. The number of partitions of input parameters should be limited to a maximal number of allowed fuzzy sets in corresponding perception frames, while number of an output partitions is determined by the significance of an output precision in the problem that is being solved.

Process of clustering is divided into two steps:

1) Creating and/or assigning a training pair to a cluster.
2) Calculating cluster output index using associated training pairs.

Influence of a training pair output value to a cluster output value (and therewith index) is proportional to the distance of the training pair from the cluster center.

3.2 Clusters Analysis for Perception Frames Partitioning

The purpose of the analysis done within this phase is disposure of fuzzy set centers that, gathered, define perception frames of input parameters. As an input for these activities we use clusters obtained from the previous phase.

A cluster is defined by the coordinates that position cluster in the input parameter space, and with adequate cluster value represented by the cluster output index. A cluster set can be represented as a two-dimensional matrix in which each row represents a single cluster, while columns define cluster coordinates and output index.

A process, that aims to determine all characteristic points in parameter input space, is based on analyses of all functions that describe output value dependence from a single input parameter. In fig. 4 these functions are shown for a problem of two input and one output parameter.

In order to approximate single input function using fuzzy neural network based on perception frames we should assign fuzzy set for each local minimum or maximum including the starting and the ending point. In addition, fuzzy sets will be needed for eventual "slope" areas of a function between local minimums and maximums. The both kind of characteristic points can be found from absolute value of a function derivate. They correspond to local minimums (including the starting and ending point). Therefore, set of characteristic points (X_{FSc}), i.e. centers of fuzzy sets, is defined by expression (1).

$$g(x)=\begin{cases}\left|\dfrac{df(x)}{dx}\right|, & x>x_{\min}\wedge x<x_{\max},\\ 0, & x\le x_{\min}\vee x\ge x_{\max}\end{cases} \quad (1)$$
$$(\forall x\in[x_{\min},x_{\max}])(\exists\varepsilon>0)$$
$$\left[x\in X_{FSc}\Leftrightarrow g(x)<g(x-\varepsilon)\wedge g(x)<g(x+\varepsilon)\right].$$

The following example illustrates how established criteria are used for construction of a parameter perception frame. Function that is being analyzed, so as its derivate are (see fig. 5):

$$f(x)=\frac{x^2}{2}+\sin\left(\frac{\pi}{2}x\right)-3,\quad -3.5\le x\le 3.5;$$
$$\frac{df(x)}{dx}=x+\frac{\pi}{2}\cos\left(\frac{\pi}{2}x\right); \quad (2)$$
$$X_{FSc}\in\{-3.5,-0.7,1.75,3.5\}.$$

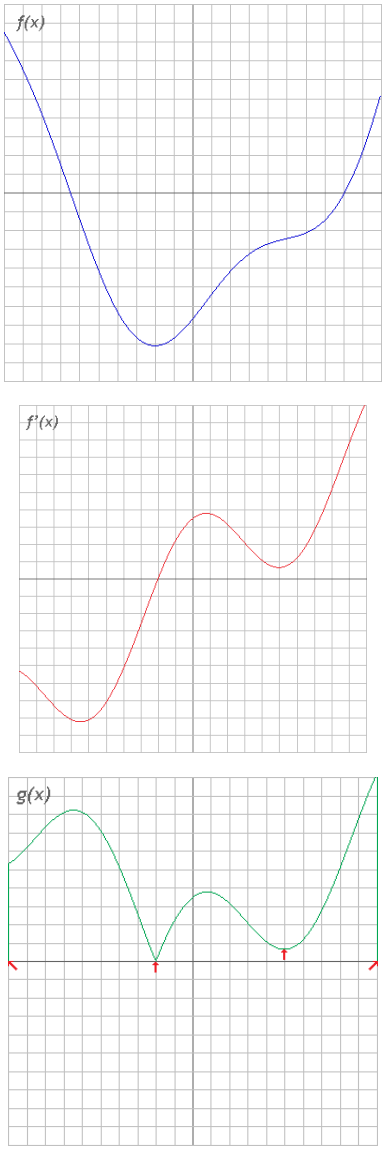


Fig. 5. Illustration of perception frame fuzzy set centers detection

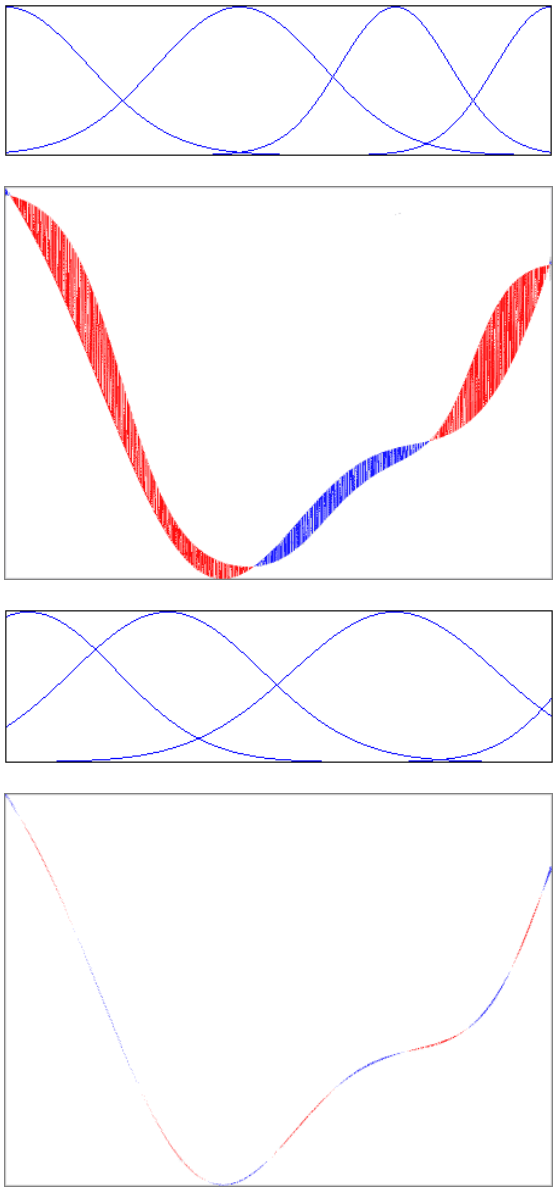


Fig. 6. Initial perception frame (a) and difference between expected and obtained output (b); Perception frame (c) and difference (d) after training for adaptation.

3.3 Setting initial fuzzy set widths

In our case, a fuzzy neural network perception frame of an input parameter is consisted of fuzzy sets defined using Gaussian-type functions, i.e. using centers and widths of Gaussian-type functions. After founding of perception frames centers of partition in previous phase, corresponding fuzzy sets are generated by assigning widths to those centers. Initial fuzzy set width is calculated as maximum of distances between two nearest fuzzy set neighbors. This principle of determining fuzzy set widths ensures full covering of an input parameter perception frame. In fig. 6.a, we have initial perception frame generated for the previous example.

3.4 Generation of fuzzy rules

Rules generation process relies on a principle of rule activation using data from a training set. For each pair of inputs and expected output, having in mind generated perception frames of input parameters, the best firing combination of fuzzy sets is found. This combination of perception frame fuzzy sets defines a rule that is to be activated. If the found combination signifies to a pre-activated rule only consequence value for that rule is modified.

Result after this phase of presented algorithm is complete fuzzy neural network with initial parameter values. This network is then fine tuned using gradient descent method and modified backpropagation algorithm.

3.5 Training for adaptation using gradient descent method

Further adjustment of a generated fuzzy neural network is done using technique of neural network supervised learning, previously described – gradient descent method (illustration in fig. 6). Training for adaptation using gradient descent method of FNN based on perception frames is described in [10].

To derive expressions for training using gradient descent method we will consider a FNN based on perception frames with m inputs (x_j, j=1,...,m) and single output (y). Perception frame for j^{th} input has n_j fuzzy sets, so it is possible to have $k = n_1 \times n_2 \times ... \times n_m$ fuzzy rules. The consequence of the rule is fuzzy singleton represented by one numerical value (w_i). Output of the network can be evaluated as:

$$y = \frac{\sum_{i1=1}^{n1}\sum_{i2=1}^{n2}\cdots\sum_{im=1}^{nm}\lambda_{i1,i2,\ldots,im}\cdot w_{i1,i2,\ldots,im}}{\sum_{i1=1}^{n1}\sum_{i2=1}^{n2}\cdots\sum_{im=1}^{nm}\lambda_{i1,i2,\ldots,im}},$$

$$\lambda_{i1,i2,\ldots,im} = \mu_{1,i1}(x_1)\cdot\mu_{2,i2}(x_2)\cdot\ldots\cdot\mu_{m,im}(x_m) = \prod_{j=1}^{m}\mu_{j,ij}(x_j), \tag{3}$$

$$\mu_{j,ij}(x_j) = \mu_{j,ij}^{G}(x_j) = \exp\left(\frac{(x_j - a_{j,ij})^2}{b_{j,ij}^2}\right)$$

To apply supervised learning on this network it is necessary to define an error function. Modification of parameters w_i, a_{ji} and b_{ji} require evaluation of the error function partial derivations for each parameter type. Expressions for modification of fuzzy singleton consequences, fuzzy set centers and widths are:

$$\varepsilon = \frac{1}{2}(y-t)^2,$$

$$w_{i1,\ldots,im}(t+1) = w_{i1,\ldots,im}(t) - \eta_w\frac{\partial\varepsilon}{\partial w_{i1,\ldots,im}},\ i_p = 1,\ldots,n_p\ (p=1,\ldots,m);$$

$$a_{j,ij}(t+1) = a_{j,ij}(t) - \eta_a\frac{\partial\varepsilon}{\partial a_{j,ij}},\ j=1,\ldots,m;\ \ i_j = 1,\ldots,n_j; \tag{4}$$

$$b_{j,ij}(t+1) = b_{j,ij}(t) - \eta_b\frac{\partial\varepsilon}{\partial b_{j,ij}},\ j=1,\ldots,m;\ \ i_j = 1,\ldots,n_j;$$

where t is an expected output of the controller, while coefficients η_w, η_a and η_b determines speed of a corresponding parameters training.

Expressions for calculation of the error function partial derivations, used for weights modification, can be evaluated as:

$$\frac{\partial \varepsilon}{\partial w_{i1,\ldots,im}} = \frac{\partial \varepsilon}{\partial y} \cdot \frac{\partial y}{\partial w_{i1,\ldots,im}} = \frac{(y-t)\lambda_{i1,\ldots,im}}{\sum_{s1,\ldots,sm} \lambda_{s1,\ldots,sm}}$$

$$\begin{aligned}
\frac{\partial \varepsilon}{\partial a_{j,i_j}} &= \frac{\partial \varepsilon}{\partial y} \cdot \sum_{p_1=1}^{n_1} \cdots \sum_{p_{j-1}=1}^{n_{j-1}} \sum_{p_{j+1}=1}^{n_{j+1}} \cdots \sum_{p_m=1}^{n_m} \left(\frac{\partial y}{\partial \lambda_{p1,\ldots,ij,\ldots,pm}} \cdot \frac{\partial \lambda_{p1,\ldots,ij,\ldots,pm}}{\partial \mu_{j,i_j}} \right) \cdot \frac{\partial \mu_{j,i_j}}{\partial a_{j,i_j}} \\
&= \frac{(y-t)}{\mu_{j,i_j} \sum_{s1,\ldots,sm} \lambda_{s1,\ldots,sm}} \cdot \sum_{p_1,\ldots,p_{j-1},p_{j+1},\ldots,p_m} \left((w_{p1,\ldots,ij,\ldots,pm} - y) \cdot \lambda_{p1,\ldots,ij,\ldots,pm} \right) \cdot \frac{\partial \mu_{j,i_j}}{\partial a_{j,i_j}} \\
&= 2 \cdot \frac{(y-t) \cdot (x_j - a_{j,i_j})}{b_{j,i_j}^2 \sum_{s1,\ldots,sm} \lambda_{s1,\ldots,sm}} \cdot \sum_{p_1,\ldots,p_{j-1},p_{j+1},\ldots,p_m} \left((w_{p1,\ldots,ij,\ldots,pm} - y) \cdot \lambda_{p1,\ldots,ij,\ldots,pm} \right)
\end{aligned} \tag{5}$$

$$\begin{aligned}
\frac{\partial \varepsilon}{\partial b_{j,i_j}} &= \frac{\partial \varepsilon}{\partial y} \cdot \sum_{p_1=1}^{n_1} \cdots \sum_{p_{j-1}=1}^{n_{j-1}} \sum_{p_{j+1}=1}^{n_{j+1}} \cdots \sum_{p_m=1}^{n_m} \left(\frac{\partial y}{\partial \lambda_{p1,\ldots,ij,\ldots,pm}} \cdot \frac{\partial \lambda_{p1,\ldots,ij,\ldots,pm}}{\partial \mu_{j,i_j}} \right) \cdot \frac{\partial \mu_{j,i_j}}{\partial b_{j,i_j}} \\
&= \frac{(y-t)}{\mu_{j,i_j} \sum_{s1,\ldots,sm} \lambda_{s1,\ldots,sm}} \cdot \sum_{p_1,\ldots,p_{j-1},p_{j+1},\ldots,p_m} \left((w_{p1,\ldots,ij,\ldots,pm} - y) \cdot \lambda_{p1,\ldots,ij,\ldots,pm} \right) \cdot \frac{\partial \mu_{j,i_j}}{\partial b_{j,i_j}} \\
&= 2 \cdot \frac{(y-t) \cdot (x_j - a_{j,i_j})^2}{b_{j,i_j}^3 \sum_{s1,\ldots,sm} \lambda_{s1,\ldots,sm}} \cdot \sum_{p_1,\ldots,p_{j-1},p_{j+1},\ldots,p_m} \left((w_{p1,\ldots,ij,\ldots,pm} - y) \cdot \lambda_{p1,\ldots,ij,\ldots,pm} \right)
\end{aligned}$$

In the case of implementation of fuzzy rule premises using triangular-type fuzzy membership functions, partial derivations, used for fuzzy set centers and widths training, becomes:

$$\mu_{j,ij}(x_j) = \mu_{j,ij}^T(x_j) = \begin{cases} 1 - \dfrac{\left| x_j - a_{j,ij} \right|}{b_{j,ij}} & , \; a_{j,ij} - b_{j,ij} \le x_j \le a_{j,ij} + b_{j,ij} \\ 0 & , \text{ in other cases} \end{cases}$$

$$\begin{aligned}
\frac{\partial \varepsilon}{\partial a_{j,i_j}} &= \frac{(y-t) \cdot \operatorname{sgn}(x_j - a_{j,i_j})}{\mu_{j,i_j} \cdot b_{j,i_j} \cdot \sum_{s1,\ldots,sm} \lambda_{s1,\ldots,sm}} \cdot \sum_{p_1,\ldots,p_{j-1},p_{j+1},\ldots,p_m} \left((w_{p1,\ldots,ij,\ldots,pm} - y) \cdot \lambda_{p1,\ldots,ij,\ldots,pm} \right) \\
\frac{\partial \varepsilon}{\partial b_{j,i_j}} &= \frac{(y-t) \cdot \left| x_j - a_{j,i_j} \right|}{\mu_{j,i_j} \cdot b_{j,i_j}^2 \cdot \sum_{s1,\ldots,sm} \lambda_{s1,\ldots,sm}} \cdot \sum_{p_1,\ldots,p_{j-1},p_{j+1},\ldots,p_m} \left((w_{p1,\ldots,ij,\ldots,pm} - y) \cdot \lambda_{p1,\ldots,ij,\ldots,pm} \right)
\end{aligned} \tag{6}$$

These expressions are used for implementation of a modified backpropagation algorithm applicable to fuzzy neural networks. The algorithm was already implemented and tested in our simulation and training environment *FNSim* as described in [12].

5 Test Results

To evaluate proposed algorithm for automatic generation of FNNs based on perception frames we employ it in solving the function approximation problem. This choice of problematic was determinate because an ability to easily generate different training sets. Furthermore, using functions with two input parameters enables intuitive validation of a generation and training process trough three-dimensional representation of the training set, achieved results and errors.

For testing purposes we implemented proposed algorithm and incorporated it in previously developed *FNSim* [12] simulation and training environment.

Testing was done using datasets generated for three different functions: sinusoidal, square type and Gaussian type (see fig. 7, 8, and 9). As input parameters for the clustering phase of the algorithm we used 7 partitions per input dimensions and 10 levels for output indexing. Achieved results are illustrated in figures 7, 8 and 9. Each of these figures shows three-dimensional diagrams with expected output values (training set), initial output values after FNN generation, and output values after training. They also show initial and trained fuzzy set distribution per perception frames of input parameters.

Initial results achieved for square type and sinusoidal function are very satisfying and show little difference from expected output. Test with Gaussian type function shows worse result in FNN generation, but it is fixed with training (see fig. 9). The reason for that is the nature of selected function that is completely inappropriate for FNN based on perception frames.

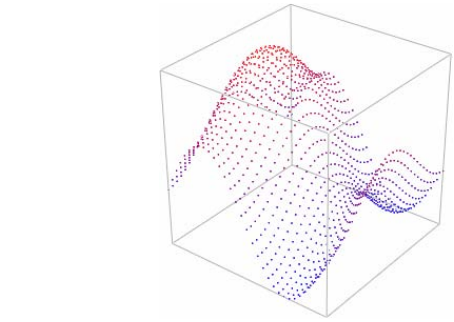
4 Conclusion

This paper introduces an algorithm for automatic generation of fuzzy neural networks based on perception frames. Targeted type of FNN use simplified fuzzy singleton reasoning method, while reliance on perception frames for partitioning input parameters space gives the ability to easily describe knowledge gathered by the FNN using fuzzy-linguistic rules.

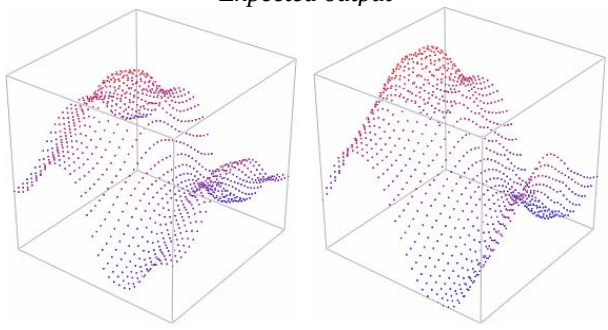
Presented algorithm supports generation of FNN in terms of perception frames and rule identification, while training for adaptation relies on gradient decent method, i.e. modified backpropagation algorithm.

Generation of initial FNN is organized in four algorithmic steps:

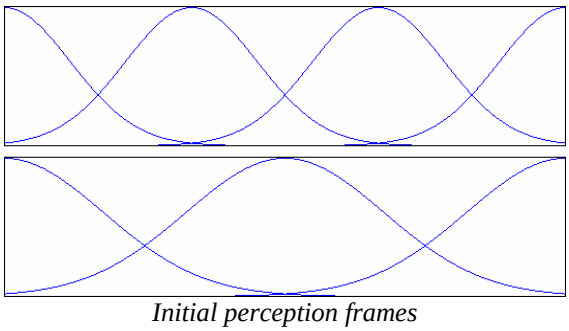
1. Clustering of an input training set in order to simplify a training set, and to filter the data of an eventual noise.
2. Determining fuzzy set number and centers in perception frames of input parameters based on analyses of all functions that describe output value dependence from a single input parameter.
3. Setting initial widths of perception frames fuzzy sets as maximum of distances between two nearest fuzzy set neighbors.
4. Generation of rules in terms of activation and consequence initialization using training set data.



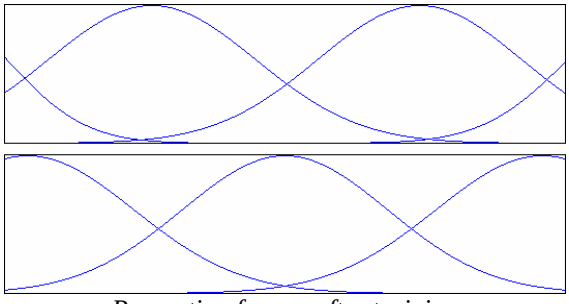
Expected output



Initial output *Output after training*



Initial perception frames



Perception frames after training

Fig. 7. Test results for sinusoidal function

Generation of FNNs that solve two-dimensional function approximation problem was used to evaluate presented approach. Although, test results presented in previous section are very satisfying, they are heavily dependent upon selected clustering parameters (number of clusters per input and output parameters) and training set density and distribution over input space. Future work should address these problems.

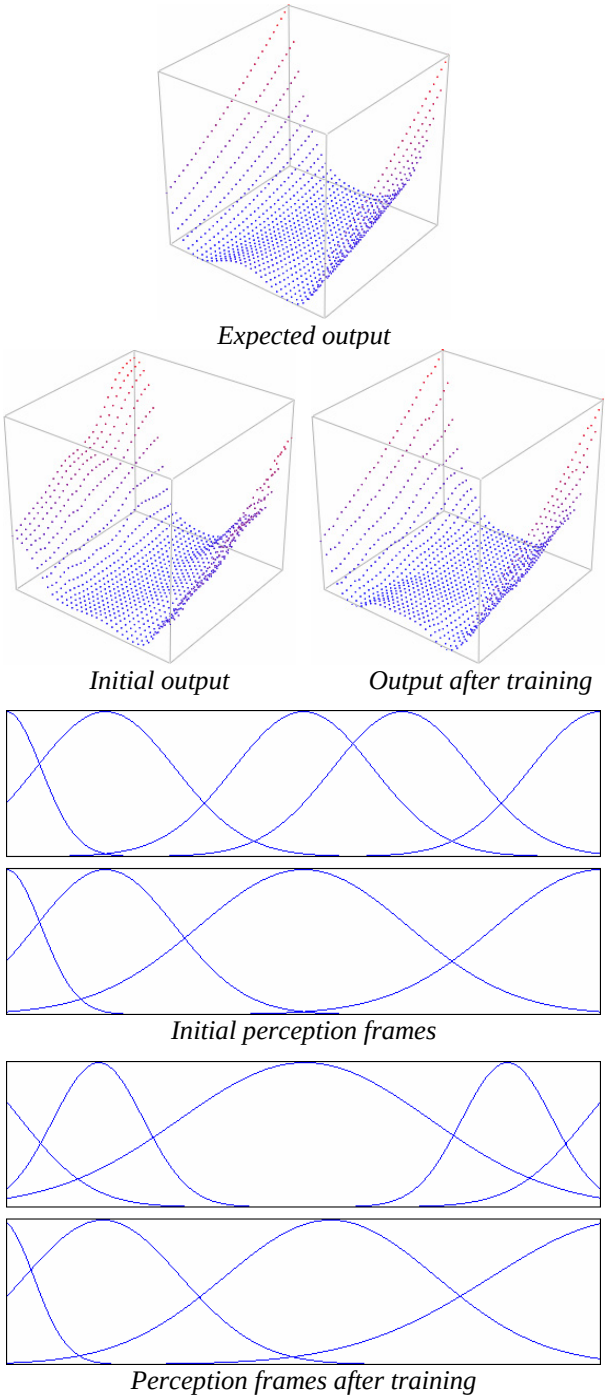


Fig. 8. Test results for square type function

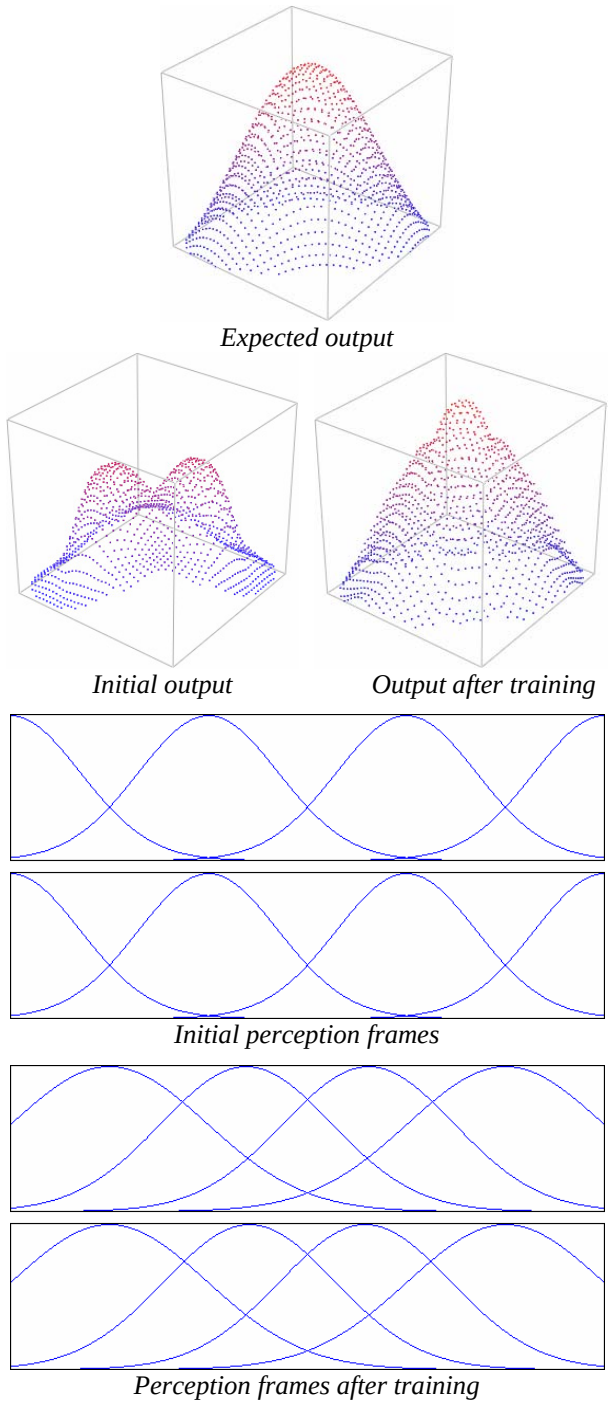


Fig. 9. Test results for Gaussian type function

References:

[1] Yuan, Y. and Suarga, S., "On the Integration of Neural Networks and Fuzzy Logic Systems", *Internat. Conf. on Systems, Man and Cybernetics*, Vancouver, Canada, Oct. 1995, pp. 452-456.

[2] Milenković, S., *Veštačke neuronske mreže*, Zadužbina Andrejević, Beograd, 1997.

[3] Subašić, P., *Fazi logika i neuronske mreže*, Tehnička knjiga, Beograd, 1997.

[4] Deplanques, P., Vaija, P. and Zapata, R., "Fuzzy Neural Networks: A Backpropagation Algorithm Specific to the Controller of Sugeno", *Internat. Conf. on Systems, Man and Cybernetics*, Vancouver, Canada, Oct. 1995, pp. 2052-2056.

[5] Mastorakis N.E., General Fuzzy Systems as extensions of the Takagi-Sugeno methodology, *WSEAS Transactions on Systems*, Issue 2, Vol. 3, April 2004, pp.795-800.

[6] Mastorakis N.E., Modeling Dynamical Systems via the Takagi-Sugeno Fuzzy Model, *WSEAS Transactions on Systems*, Issue 2, Vol. 3, April 2004, pp.668-676.

[7] Wang, Y. and Rong, G., A self-organizing neural-network-based fuzzy system, *Fuzzy Sets and Systems*, Vol. 103, 1999, pp. 1-11.

[8] Shi, Y. and Mizumoto, M., Some consideration on conventional neuro-fuzzy learning algorithms by gradient descent method, *Fuzzy Sets and Systems*, Vol. 112, 2000, pp. 51-63.

[9] Yang, Y., Xu, X. and Zang, W., Design neural networks based fuzzy logic, *Fuzzy Sets and Systems*, Vol. 114, 2000, pp. 325-328.

[10] Shi, Y. and Mizumoto, M., A new approach of neuro-fuzzy learning algorithm for tuning fuzzy rules, *Fuzzy Sets and Systems*, Vol. 112, 2000, pp. 99-116.

[11] Wang H., Cheng P., Fault Diagnosis for a Rolling Bearing Used in a Reciprocating Machine by Adaptive Filtering Technique and Fuzzy Neural Network, *WSEAS Transactions on Systems*, Issue 1, Vol. 7, January 2008, pp. 1-6.

[12] Milosavljević, A., Stoimenov, L. and Tošić, Ž., "Fuzzy Neural Networks Analyses through Simulation and Training", *Internat. Conf. on Computational Intelligence for Modelling, Control and Automation - CIMCA 2003*, Vienna, Austria, Feb. 12-14, 2003, pp. 916-925.