

Training Neurofuzzy Networks with Participatory Learning

Michel Hell
UNICAMP
FEEC - DCA
Campinas, SP
Brazil, 13083-970
mbhell@dca.fee.unicamp.br

Rosângela Ballini
UNICAMP
IE - DTE
Campinas, SP
Brazil, 13083-970
ballini@eco.unicamp.br

Pyramo Costa Jr.
PUC-MG
PPGEE
Belo Horizonte, MG
Brazil, 30535-610
pyramo@pucminas.br

Fernando Gomide
UNICAMP
FEEC - DCA
Campinas, SP
Brazil, 13083-970
gomide@dca.fee.unicamp.br

Abstract

This paper introduces a new approach to adjust a class of neurofuzzy networks based on the idea of participatory learning. Participatory learning is a mean to learn and revise beliefs based on what is already known or believed. The performance of the approach is verified with the Box and Jenkins gas furnace modeling problem, and with a short-term load forecasting problem using actual data. Comparisons with alternative training procedures suggested in the literature are included to shown the effectiveness of participatory learning to train neurofuzzy networks.

Keywords: Participatory Learning, Fuzzy Systems, Neurofuzzy Networks.

1 Introduction

Neurofuzzy systems have been widely used in different areas of science and engineering, including medicine, economics, fuzzy mathematics, game theory, systems modeling to mention a few.

As part of computational intelligence, neurofuzzy systems combine two major paradigms, neural networks and fuzzy systems. The combination results in systems that integrate the representation richness of fuzzy systems with the learning ability of neural networks. The central idea behind neurofuzzy systems is the capability to tune fuzzy systems using learning procedures and data. Suitable methodologies and learning algorithms are among the most important issues when designing neurofuzzy systems [1].

In the past years several methods to train neurofuzzy systems, especially neurofuzzy networks, have been proposed. In general these methods are based on gradient descent and/or reinforcement approaches [1, 2, 3, 4].

These techniques produce satisfactory results in many cases but currently there is no consensus on which one is the best since they depend heavily on the data set characteristics [5].

In 1990 a new learning approach, namely Participatory Learning (PL), was introduced by Yager as a model of learning that captures many of the salient features of human learning [6]. This approach assumes that learning and beliefs about an environment depend on what the system already knows about the environment [7].

Recently, the participatory learning paradigm was used to develop an efficient unsupervised fuzzy cluster algorithm [5] and to find rule base structures in adaptive, evolving fuzzy modeling procedures [8].

Alternative methods, such as Bayesian analysis [9], have been developed to combine current belief with new knowledge about the environment coming from observations. In this work we adopt an approach based on the assumption that probability density functions are not easily available or computable, what precludes the use of Bayesian approaches.

More precisely, this paper suggests the use of PL idea to develop a new training procedure for hybrid neurofuzzy networks. The procedure does not dependent on the data set because the current knowledge about the environment is part of the learning process itself and influences the way in which new observations are used for learning [5].

The paper is organized as follows. Next section reviews the main ideas and concepts of participatory learning. Section three presents the neurofuzzy network structure. Session 4 details the training procedure suggested in the paper. Session 5 summarizes simulation results and comparison studies. Section 6 concludes the paper summarizing its contributions and issues for further investigation.

2 Participatory Learning

The main characteristic of participatory learning is that an exogenous observation impact in causing learning or belief revision depends on its compatibility with the current system belief. In particular, observations conflicting with the current belief are discounted [6].

Let $v \in [0, 1]^n$ be a variable that represents the belief of a system. The aim of the participatory learning is to learn the value of this variable based on a sequence of observations $x^k \in [0, 1]^n$ that encodes the knowledge about the value of the variable v . In this sense x^k is a manifestation of value of v in the k -th observation. Thus we use the vector x^k as a means to learn valuations of v . The learning process is participatory if the usefulness of each observation x^k in contributing to the learning process depends upon its acceptance by the current estimate of the values of v as being valid observation [6]. Participatory learning means that, to be relevant for the learning process, x^k must be close to v^k . A mechanism for updating the current beliefs of v is a smoothing like algorithm:

$$v^{k+1} = v^k + \alpha \rho_k (x^k - v^k) \quad (1)$$

where $k = 1, \dots, P$, and P is the number of observations; v^{k+1} is the new system belief; v^k is the current belief; x^k is the current observation; $\alpha \in [0, 1]$ is the learning rate; and $\rho_k \in [0, 1]$ is the compatibility degree between x^k and v^k . As an example, ρ_k can be computed as $\rho_k = 1 - \frac{1}{n} \sum_{i=1}^n d_i^k$ where $d_i^k = |x_i^k - v_i^k|$.

Notice that, if $\rho_k = 0$ (observation too far from current belief) we get from (1) that $v^{k+1} = v^k$ and the system is maximally closed to learn. On the other hand, if $\rho_k = 1$ we see from (1) that $v^{k+1} = v^k + \alpha(x^k - v^k)$, therefore the system is maximally open for learning. However, in order for ρ_k to equal 1 it must be the case that $x_i^k - v_i^k = 0$ for all i . This condition implies that $v^k = x^k$ and hence we get $v^{k+1} = v^k$, which implies that no learning occurs [10].

One concern that can be raised about the described learning procedure is that it ignores the situation when a stream of low ρ_k 's arises during a long period of time. In this case, the system should become more open to learning because it may be the case that the current belief is wrong, not the new observations. In [6] Yager proposes a mechanism that monitors the compatibility of the current beliefs with the observations. This information is translated into an arousal index used to influence the learning process, as shown in Figure 1. The higher the arousal rate, the less confident is the system with the current belief, and conflicting observations become important to update the beliefs. Let us denote $a_k \in [0, 1]$ as the arousal index. The higher a_k the more aroused the system will be. The dynamics of

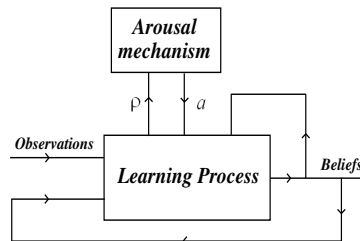


Figure 1: Participatory Learning with Arousal

the arousal index adaptation is specified as follows:

$$a_{k+1} = a_k + \lambda((1 - \rho_{k+1}) - a_k) \quad (2)$$

where $\lambda \in [0, 1]$ is a constant that controls the rate of change arousal; the closer λ is to one, the faster the system is to sense compatibility variations. The arousal index can be seen as the complement of the confidence in the belief structure currently held.

Expression (1) can be rewritten to incorporate the arousal mechanism as follows:

$$v^{k+1} = v^k + \alpha \rho_k^{1-a_k} (x^k - v^k) \quad (3)$$

As it can be noted in (3), while ρ_k measure how much the system changes its credibility in its own beliefs, the arousal index a_k acts as a critic to remind when the current belief should be modified in front of new evidences. Moreover, apart from the term $\rho_k^{1-a_k}$ Equation (3) can be viewed as a standard learning rule used to training neural networks. However, in the presence of spurious samples in training data set, a learning rule without the term $\rho_k^{1-a_k}$ may not converge or converge to an incorrect value.

3 Neurofuzzy Network Model

This section introduces the complete structure of the hybrid neurofuzzy network (RNF), which is composed by two parts. The first part has an input and two hidden layers and assembles a fuzzy inference system. The second is a classic neural network whose purpose is to aggregate the outputs of the fuzzy inference system. Figure 2 shows a neurofuzzy network with n inputs and m outputs. The input layer supplies the inputs to the neurons in the next layer. The first hidden layer consists of neurons whose activation functions are membership functions of fuzzy sets that form the input space partition.

For each dimension x_i^k of a n -dimensional input vector x^k there are N_i fuzzy sets $A_i^{\lambda_i}$, $\lambda_i = 1, \dots, N_i$ whose membership functions are activation functions of corresponding input layer neurons. The variable k is the

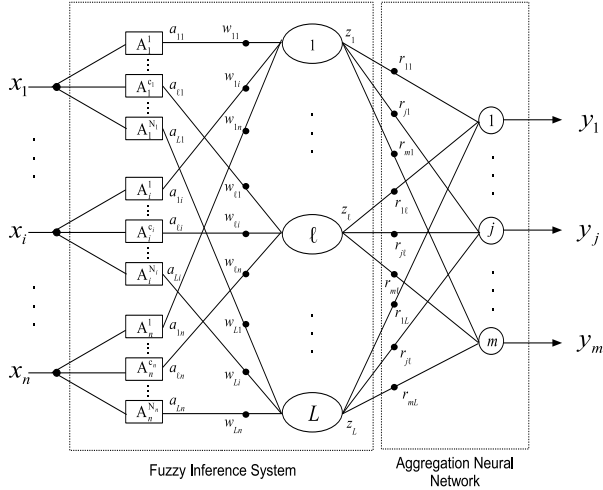


Figure 2: Hybrid Neurofuzzy Network Structure

discrete-time, that is, $k = 1, 2, \dots$, but it will be omitted in the rest of the paper to simplify notation. Thus, the outputs of the first hidden layer are the membership degrees associated with the input values, that is, $a_{\ell i} = \mu_{A_i^{\lambda_i}}(x_i)$, $i = 1, \dots, n$ and $\ell = 1, \dots, L$; where L is the number of neurons in the second hidden layer.

The neurons of the second hidden layer are fuzzy set-based neurons called *or* neurons (Figure 3) whose inputs $a_{\ell i}$ are weighted by $w_{\ell i}$ where $w_{\ell i}$ corresponding to the credibility of the input $a_{\ell i}$. The neurons use *s*-norms to process the inputs. For different instances of *s*-norms see [2] and references therein.

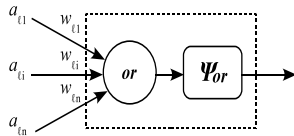


Figure 3: Fuzzy Set *or* Neurons

Fuzzy set-based neurons assume membership degrees $a_{\ell i}$ inputs and weights $w_{\ell i}$ in $[0, 1]$. Therefore they perform nonlinear mappings $[0, 1]^n \rightarrow [0, 1]$. The activation function ψ_{or} of a *or* neuron is, in general, a nonlinear function, but here we assume ψ_{or} to be the identity function $\psi_{or}(u) = u$.

It is easy to see that the network structure embeds a set of *if-then* rules $R = \{R_k, k = 1, \dots, L\}$ of the form:

- R_k : If (x_1 is $A_1^{\lambda_1}$ with credibility $w_{\ell 1}$)...
or (x_i is $A_i^{\lambda_i}$ with credibility $w_{\ell i}$)...
or (x_n is $A_n^{\lambda_n}$ with credibility $w_{\ell n}$)
 Then, z is z_ℓ .

where

$$z_\ell = \mathbf{S}_{i=1}^n (w_{\ell i} \mathbf{t} a_{\ell i}) .$$

Therefore, there is a close correspondence between the structure of the first part of the neurofuzzy network and a set of fuzzy rules or, alternatively, a fuzzy rule base. In addition, the processing scheme induced by the network structure agrees with the principles of fuzzy set theory and approximate reasoning [2].

As mentioned above, the second part is a classical neural network composed by conventional neurons whose output y_j is an aggregation of the inputs z_ℓ and the weights $r_{j\ell}$, $\ell = 1, \dots, L$, $j = 1, \dots, m$.

The neurofuzzy network processing can be summarized as follows:

1. N_i is the number of fuzzy sets that composes the partition of i -th input;
2. $a_{\ell i} = \mu_{A_i^{\lambda_i}}(x_i)$ is the membership degree of x_i in the fuzzy set $A_i^{\lambda_i}$, where $a_{\ell i}$ is the i -th input of neuron ℓ of the second hidden layer;
3. z_ℓ is the output of the ℓ -th neuron of the second hidden layer:

$$z_\ell = \mathbf{S}_{i=1}^n (w_{\ell i} \mathbf{t} a_{\ell i}) \quad (4)$$

4. y_j is the output of j -th nonlinear neuron of the output layer given by:

$$y_j = f(u_j) = f\left(\sum_{k=1}^L (v_{jk} z_k)\right) \quad (5)$$

where, $f : \mathfrak{R}^L \rightarrow [0, 1]$ is a nonlinear, monotonically increasing function. In this paper, we use the logistic function $f(u_j) = 1/(1 + \exp(-u_j))$;

5. $w_{\ell i}$ is the weight between the ℓ -th *or*-neuron and the i -th neuron of the first hidden layer;
6. $r_{j\ell}$ is the weight between the output y_j of the network and the ℓ -th neuron of the second hidden layer;

The neurofuzzy network architecture is very flexible because it allows appropriate triangular norms to be chosen and offers the possibility to extract linguistic fuzzy rules directly from its topology.

4 Training Procedure

The training procedure suggested in this paper involves three main phases. The first phase uses the fuzzy c-means algorithm to granulate the input space. The next phase use the gradient descent to update the

weights related to the classic neural network ($r_{j\ell}$'s). The last phase use the participatory learning paradigm summarized in section 2 to update the weights connected to the *or* neurons ($w_{\ell i}$'s). The essential implementation steps are detailed next.

4.1 Generation of Membership Functions

To generate the membership functions of the first hidden layer of the network shown in Figure 2 the c-means clustering algorithm is used. Information about input and output spaces are included in the cluster process. The modal values of the membership functions are the projection of the clusters centers in its respective universes. We assume Gaussian-shaped function as membership functions in the first hidden layer neurons. The dispersions of the Gaussians are adjusted to form a cognition frame of the input space [2].

4.2 Weight Updating

The first step in the weight updating process is to estimate the network output $\hat{y} \in [0, 1]^m$ for a given input $x \in [0, 1]^n$. This corresponds to the fuzzification of the input pattern, and successive computation the outputs of the remaining neurons layers using Equations (4) and (5). Next, the learning process aims to minimizing an error measure between the actual network output and a desired output for each input pattern, that is, to minimize

$$e = \frac{1}{2} \sum_{j=0}^m (y_j - \hat{y}_j)^2 \quad (6)$$

where \hat{y}_j is the value of the output unit j and y_j is the desired value of the output unit j .

We update the output layer weights using a gradient descent method, that is

$$\Delta r_{j\ell} = \eta (y_j - \hat{y}_j) f'(u_j) z_{\ell} \quad (7)$$

where $f'(u_j) = f(u_j)(1 - f(u_j))$ is the derivative of the activation function evaluated at u , $u_j = \sum_{\ell=1}^L (r_{j\ell} z_{\ell})$, and η is the learning rate.

The next step update the weights of the logical neurons using the participatory learning idea, as follows.

Participatory Learning Procedure

As discussed in the Section 3, the fuzzy inference system encodes in the fuzzy set-based neurons structures can be seen as a fuzzy relation $G : A \times Z \rightarrow [0, 1]$ where A and Z are two finite universes and G is any subset of the Cartesian product of these two universes. In particular, for $a^k \in A$ and $G \subset A \times Z$ we have that it is possible to compute an element $z^k \in Z$ related with a^k of the form $z^k = a^k \circ G$.

If we choose as s -norm the maximum to the *or* neurons we have that the operator ' \circ ' becomes *max-t*

(*sup-t*) and for a given pair (a^k, z^k) the relation G can be found by solving the fuzzy-relational estimation problem [2] whose solution is:

$$G = a^k T \varphi z^k \quad (8)$$

where T denote the transpose and φ is such that

$$(a_{\ell i}^k \varphi z_{\ell}^k) = \text{sup}(c \in [0, 1] \mid a_{\ell i}^k t c \leq z_{\ell}^k) \quad (9)$$

We can see that the relational matrix $G = [g_{\ell i}]$ have strong relation with the weight matrix $W = [w_{\ell i}]$. In fact, for a given pattern $[x_k, y_k]$ and a matrix $R = [r_{j\ell}]$ of output layer weight's the better approximation of the mapping $f : x_k \rightarrow y_k$ performed by the network shown of Figure 2 occurs when $W \equiv G$.

To find the relation G we need the output of logical layer neurons z_k values. This can be estimated from y_k and R solving the constrained linear least-squares problem:

$$\min_{z^k} \left\{ \frac{1}{2} |Rz^k - f^{-1}(y^k)|^2 \right\} \quad \text{subject to } 0 \leq z^k \leq 1 \quad (10)$$

where $f^{-1}(\cdot)$ is the inverse of function f in (5).

Next, we compute the fuzzy relation G using (9).

Once G is found we can update weights $w_{\ell i}$'s based on participatory learning paradigm as follows.

$$\Delta W^k = \alpha (\rho^k)^{1-a_k} (G^k - W^k) \quad (11)$$

where α is the learning rate; $k = 1, \dots, P$ and P is the number of training patterns; a_k is calculated according (2), and

$$\rho^k = 1 - \frac{1}{Ln} \sum_{\ell=1}^L \sum_{i=1}^n |g_{\ell i}^k - w_{\ell i}^k|. \quad (12)$$

The procedure to train the neurofuzzy networks using PL approach (NFPL) proposed in this paper can be summarized in the pseudo code presented in Figure 4.

5 Experimental Results

To evaluate the performance of the learning algorithm introduced in this paper, we compare our approach with alternative models suggested in the literature. The examples given here include nonlinear system identification and time series prediction problems. We adopt the maximum s -norm and the algebraic product as the t -norm in the *or* neurons.

Box and Jenkins Gas Furnace

The Box and Jenkins system identification is a well-known benchmark problem. Identification uses of 290 pairs of input/output data taken from a laboratory furnace [11]. Each data sample consists of the methane flow rate, the process input variable, x^k , and the CO_2 concentration in off gas, the process output, y^k . This is a dynamical process with one input x^k and one output y^k . The aim is to predict current output $y(k)$ using

Begin NFPL

```

Read Training Data;
Granularize the input space (section 4.1);
p:= n° of training patterns;
WHILE err ≥ error tolerance DO
  t:=1;
  WHILE t ≤ p DO
    Read (xt, yt);
    Estimate the network output yk (section 3);
    Compute the approximation error e using (7);
    Compute the variation of the weights rji's using (8);
    Update the rji's weights;
    t:=t+1;
  END WHILE
  t:=1;
  WHILE t ≤ p DO
    Read (xt, yt);
    Compute the zk from yk and R using (11);
    Compute the fuzzy relation G using (9);
    Determine the compatibility degree ρk using (13);
    Determine the arousal index using (3);
    Compute the variation of the weights wfi using (12);
    Update the wfi's weights;
    t:=t+1;
  END WHILE
  Calculate the general approximation error err
END WHILE
END NFPL

```

Figure 4: NPFL Approach

past input and output values, with the lowest error. Different studies indicate that the best model structure for this system is $y^k = f(y^{k-1}, x^{k-4})$.

Figure 5 depicts the model output obtained by the NFPL approach. Comparison results with previous fuzzy/neurofuzzy approaches that works with the same inputs is given in Table 1. We notice that NFPL perform as well as the one of the best model.

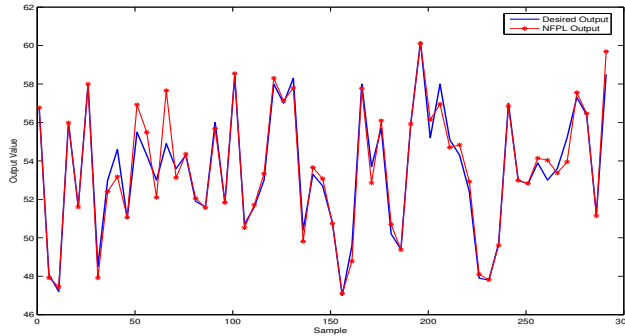


Figure 5: Box and Jenkins Gas Furnace Identification

Load Forecasting

An application example that shows the effectiveness of the participatory learning in neurooffuzy training is the short-term load forecasting. The task load forecasting is to accurately predict the 24 hourly loads of the next operation day.

We use a training data set composed by hourly loads from May of 2000 to February of 2001 of an electrical

Table 1: Gas Furnace Identification

Model	N° Rules	Mean Square Error
Pedrycz [12]	81	0.5656
Xu and Lu [13]	25	0.5727
Delgado [14]	4	0.4100
Yoshinari [15]	6	0.5460
Rutkowski [16]	9	0.4912
NFPL	9	0.4528

utility located at the Southeast region of Brazil. The input variables of the model are L_{h-1} and L_{h-2} , which represent the load at times $h-1$ and $h-2$, respectively.

To show the efficiency and the data-independent nature of the PL approach during training, two cases are considered. The first case (Case 1) data was collected without any measurement error. In the second case (Case 2) we introduce uniform random noise in selected samples of the training data. Four different neural network models were trained using data of the two cases, namely, a classic multilayer perceptron with 2 hidden layers with 12 and 8 neurons, respectively, trained with backpropagation algorithm (MLP), an adaptive neurofuzzy system [17] using the same number of fuzzy sets to granulate each input variable (ANFIS), and the neurofuzzy network presented in Figure 2 with 64 neurons in second hidden layer. The RNF was trained in two different ways. The first way uses participatory learning as detailed in Section 4 (NFPL). The second one adopts a non-participatory version (NF-NPL) of the learning procedure proposed in section 4, that is, we exclude the participatory with arousal term $\rho_k^{1-a_k}$ from Equation (11). The Mean Absolute Error (MAE) is used to evaluate the results.

$$MAE = \frac{1}{P} \sum_{k=1}^P |\hat{y}^k - y^k| \frac{100}{y^k} (\%) \quad (13)$$

where \hat{y}^k is the k -th predicted value, y^k is k -th real value and P is the number of values to predict. The results obtained are summarized in Table 2. Figure 6 shows the load forecast results provided by the neural models for September 12, 2001 in both, Case 1 (a) and Case 2 (b).

Table 2: Load Forecast

Model	MAE (%) - Case 1	MAE (%) - Case 2
MLP	7.20	12.95
ANFIS	5.74	11.47
NF-NPL	6.86	12.07
NFPL	3.98	4.42

As it can be seen in Figure 6 and Table 2, all neural models show comparable performance in Case 1. However, in Case 2 the participatory model improves the forecast substantially and outperforms the remaining neural models. The participatory nature of NFPL is

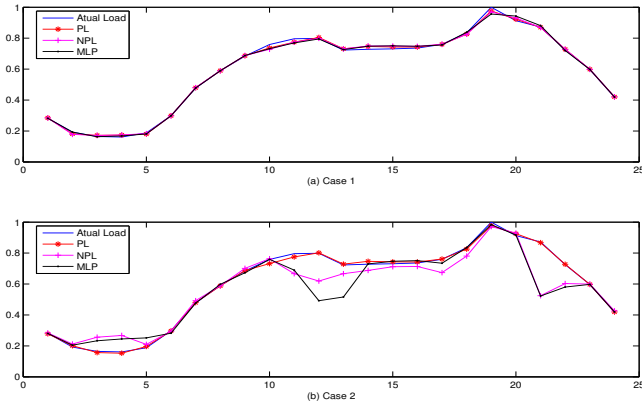


Figure 6: Short Term Load Forecasting

able to smooth the effect of noise using the compatibility measure.

6 Conclusions

In this paper we have introduced a new training procedure for a class of hybrid neurofuzzy network based on participatory learning paradigm. Experimental results show that participatory learning provide an effective alternative for supervised training of neurofuzzy networks that do not dependent of training data set and is as efficient as alternative fuzzy/neurofuzzy approaches addressed in the literature.

Acknowledgement

The authors acknowledge FAPESP, the Research Foundation of the State of São Paulo, for fellowship 03/05042-1 and grant 03/10019-9, and CNPq, the Brazilian National Research Council, for grant 304857/2006-8, and the anonymous reviewers for their constructive comments and suggestions that help to improve this paper.

References

- [1] M. Figueiredo, R. Ballini, S. Soares, M. Andrade, F. Gomide, Learning Algorithms for a class of Neurofuzzy Network and Application, *IEEE Trans. on System, Man and Cybernetics - Part C*, vol. 34, no. 3, pp. 293–301, 2004.
- [2] W. Pedrycz, F. Gomide, Fuzzy Systems Engineering: Toward Human-Centric Computing, *Wiley Interscience*, Hoboken, NJ, USA, 2007.
- [3] L. X. Wang, Adaptive Fuzzy Systems and Control, *Prentice-Hall*, NJ, USA, 1994.
- [4] C. T. Lin, C. S. Lee, Neural Fuzzy Systems, *Prentice Hall*, NJ, USA, 1996.

- [5] L. Silva, F. Gomide, R. R. Yager, Participatory Learning in Fuzzy Clustering, *Proc. 14th IEEE International Conference on Fuzzy Systems*, Reno, USA, May 2005, pp. 857–861.
- [6] R. R. Yager, A model of Participatory Learning, *IEEE Trans. on System, Man and Cybernetics*, vol. 20, no. 5, pp. 1229–1234, 1990.
- [7] R. R. Yager, Participatory Learning: A Paradigm for More Human Like Learning, *Proc. 2004 IEEE International Conference on Fuzzy Systems*, vol. 1, July 2004, pp. 79–84.
- [8] E. Lima, F. Gomide, R. Ballini, Participatory Evolving Fuzzy Modeling, *Proc. 2006 International Symposium on Evolving Fuzzy Systems*, Lake district, UK, Sept. 2006, pp. 36–41.
- [9] W. A. Wright, Bayesian approach to neural-network modeling with input uncertainty, *IEEE Trans. on Neural Networks*, vol. 10, no. 6, pp. 1261–1270, 1999.
- [10] R. R. Yager, D. P. Filev, A fuzzy logic controller view of participatory learning, *Proc. Second IEEE International Conference on Fuzzy Systems*, vol. 2, April 1993, pp. 912–917.
- [11] G. E. P. Box, G. M. Jenkins, Time Series Analysis- Forecasting and Control, 2.ed, *Holden Day*, CA, USA, 1976
- [12] W. Pedrycz, A Identification Algorithm in Fuzzy Relational Systems, *Fuzzy Sets and Systems*, vol. 13, pp. 153–167, 1984.
- [13] W. Xu, Y. Z. Lu, Fuzzy Model Identification and Self-learning for Dynamic Systems, *IEEE Trans. on System, Man and Cybernetics*, vol. SMC-17, pp. 683–689, July 1987.
- [14] M. Delgado, A. F. Gmez-Skarmeta, F. Martin, Fuzzy Clustering based Rapid Prototyping for Fuzzy Rule-based Modeling, *IEEE Trans. on Fuzzy Systems*, vol. 5, pp. 223–233, May 1997.
- [15] Y. Yoshinari, W. Pedrycz, K. Hirota, Construction of Fuzzy Models Through Clustering Techniques, *Fuzzy Sets and Systems*, vol. 54, pp. 157–165, 1993.
- [16] L. Rutkowski, and K. Cpalka, Designing and Learning of Adjustable Quasi-Triangular Norms With Applications to Neuro-Fuzzy Systems, *IEEE Trans. on Fuzzy Systems*, vol. 13, no. 1, pp. 140–151, Feb. 2005.
- [17] R. Jang, ANFIS: Adaptive network based fuzzy inference system, *IEEE Trans. on Systems Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.