International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2015): 6.391

A Method for Training a Fuzzy Constraint Network

Huy-Khoi Do[1], Thi-Xuan Tran[2], Van-Nui Nguyen[3]

[1]University of Information and Communication Technology (ICTU), Quyet Thang, Thai Nguyen, Vietnam

[2]University of Economics and Business Administration, Tan Thinh, Thai Nguyen, Vietnam

[3,*]University of Information and Communication Technology (ICTU), Quyet Thang, Thai Nguyen, Vietnam

Abstract: ***The architectures and learning procedures underlying ANFIS (Adaptive Network-based Fuzzy Inference System) and M-ANFIS (Mamdani- ANFIS) were presented and successfully applied in many fields of science areas. Based on principles of ANFIS and M-ANFIS, with the learning procedures of these two architectures, we proposed the so-called C-ANFIS (Constraint-ANFIS), being useful for training a fuzzy constraint network.***

Keywords: Fuzzy constraint network, ANFIS, M-ANFIS, C-ANFIS

1. Introduction

System modelling based on conventional mathematical tools (e.g., differential equations) is not well suited for dealing with ill-defined and uncertain systems [1-3,5,8, 9, 12]. By contrast, a fuzzy inference system employing fuzzy if-then rules can model the qualitative aspects of human knowledge and reasoning processes without employing precise quantitative analyses.

Jyh-Shing Roger Jang et al. [1, 2, 3] suggested the so-called ANFIS (Adaptive-Network-based Fuzzy Inference System), which is a fuzzy inference system implemented in the framework of adaptive networks. By using learning procedures, especially the Hybrid learning procedure, ANFIS can construct an input-output mapping based on both human knowledge (in form of fuzzy if-then rules) and stipulated input-output data pairs. In simulation, this ANFIS architecture is employed to model nonlinear functions, identify nonlinear components on-linely, all yielding remarkable results. However, these effective schemas and learning procedures are based on fuzzy if-then rules. In fact, there exist some systems and functions which are needed to model not only based on fuzzy if-then rules, but also based on some specific types of constraints. ANFIS is successful with knowledge and data pairs in form of fuzzy if-then rules, but there is very seldom schemas (or even there is no) papers mentioned about how to deal with the systems which are based on *some specific types of constraint*, included. And, hence, this is now a hot problem interested by many researchers and scholars.

This paper presented a method for training a fuzzy constraint network. Besides, we proposed the so-called C-ANFIS, which is better than ANFIS in dealing with those systems above. In particular, we also presented the learning procedure which can be applied for training a fuzzy constraint network.

2. Preliminary Knowledge

2.1. Adaptive-Network-based Fuzzy Inference System (ANFIS)

2.1.1. The Network architecture

ANFIS was firstly proposed by Roger Jang et al. [1], which can serve as a basis for constructing a set of fuzzy if-then rules with appropriate membership functions to generate the stipulated input-output pairs. ANFIS can be described briefly as follows:

For simplicity, we assume that the fuzzy inference system under consideration has two inputs x and y; one output f. Suppose that the rule base contains two fuzzy if-then rules of Tagaki's and Sugeno's type:

Rule 1: *If x is A_1 and y is B_1 then $f_1=p_1x+q_1y+r_1$*

Rule 2: *If x is A_2 and y is B_2 then $f_2=p_2x+q_2y+r_2$*

Then, the type-3 fuzzy reasoning and the corresponding equivalent ANFIS architecture (type-3 ANFIS) is shown as Figure 1:

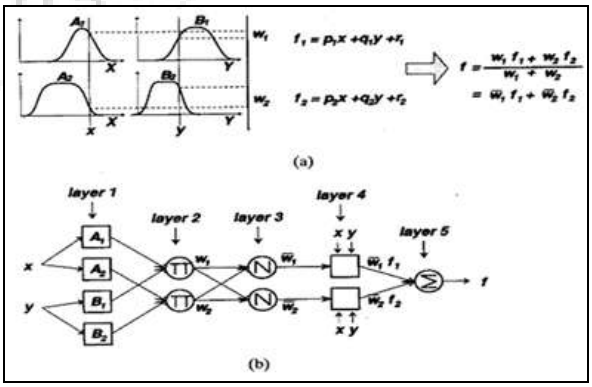

Figure 1: Type-3 fuzzy reasoning and Equivalent ANFIS

The node functions in the same layer are of the same function family as described as below:

Layer 1: Every node i in this layer is an adaptive node with a node function

$$O_{1,i} = \mu_{A_i}(x), \text{ for } i = 1, 2, \text{ or}$$

$O_{1,i} = \mu_{B_{i-2}}(y)$ *for i=3, 4.*

The membership function is chosen here is usually bell-shaped, such as:

$$\mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{d_i}\right)^2\right]^{b_i}}, \text{ or}$$

$$\mu_{A_i}(x) = \exp\left\{-\frac{1}{2}\left(\frac{x-c_i}{\sigma_i}\right)^2\right\}$$

Layer 2: Every node in this layer is a fixed node labeled **Π** which multiplies the incoming signals and sends the product out.

$$O_{2,i} = \omega_i = \mu_{A_i}(x) \times \mu_{B_i} \quad i = 1,2$$

Each node output represents the firing strength of a rule (In fact, any other T-norm operators that perform fuzzy AND can be used as the node function in this layer)

Layer 3: Every node in this layer is a circle fixed node labeled N. The i^{th} node calculates the ratio of the i^{th} rule's firing strength to the sum of all rules's firing strengths.

$$O_{3,i} = \varpi_i = \frac{\omega_i}{\omega_1 + \omega_2}; \quad i = 1,2$$ (Outputs of this layer usually be called as normalized firing strengths)

Layer 4: Every node in this layer is an adaptive node with a node function

$$O_{4,i} = \varpi_i f_i = \varpi_i (p_i x + q_i y + r_i)$$

where ϖ_i is output of layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set. Parameters in this layer will be referred to as *consequent parameters.*

Layer 5: The single node in this layer is a fixed node labeled $\sum$, which computes the overall outputs as the summation of all incoming signals.

$$O_{5,1} = overll\ output = \sum_i \varpi_i f_i = \frac{\sum \omega_i f_i}{\sum \omega_i}$$

2.1.2. Learning procedures

One of the most important thing of ANFIS is its abilities in learning and adapting by Hybrid learning procedure. The abilities of ANIFIS in learning and adapting were firstly given by Roger Jang et al. [1]. In this section, we will summary the basics of Hybrid learning of ANFIS.

From the ANFIS architecture in Figure 1, it is observed that given the values of premise parameters, the overall output can be expressed as a linear combinations of the consequent parameters. More precisely, the output in Figure 1 can be written as:

$$f = \frac{\omega_1}{\omega_1+\omega_2} f_1 + \frac{\omega_2}{\omega_1+\omega_2} f_2 = \varpi_1 f_1 + \varpi_2 f_2$$
$$= (\varpi_1 x) p_1 + (\varpi_1 y) q_1 + (\varpi_1) r_1 + (\varpi_2 x) p_2 + (\varpi_2 y) q_2 + (\varpi_2) r_2$$

which is linear in the consequent parameters (p_1, q_1, r_1, p_2, q_2, r_2). As a result, we have:

+ S = set of total parameters
+ S_1 = set of premise parameters
+ S_2 = set of consequent parameters.

Then $S = S_1 \oplus S_2$, and there exist functions H(∘) and F(∘,∘) are identify function and the function of fuzzy inference system, respectively.

Therefore, the Hybrid learning algorithm can be applied directly.

***Forward pass**:*

In the forward pass of the hybrid learning algorithm, node outputs go forward until layer 4 and the consequent are identified by the least-squares method.

When the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters.

$$f = \frac{\omega_1}{\omega_1+\omega_2} f_1 + \frac{\omega_2}{\omega_1+\omega_2} f_2 = \varpi_1 f_1 + \varpi_2 f_2$$
$$= (\varpi_1 x) p_1 + (\varpi_1 y) q_1 + (\varpi_1) r_1 + (\varpi_2 x) p_2 + (\varpi_2 y) q_2 + (\varpi_2) r_2$$
$$f = XW$$

If matrix X is invertible then

$$W = X^{-1} f$$

Otherwise, a pseudo-inverse is used to solve for W.

$$W = (X^T X)^{-1} X^T f$$

Backward pass

In the backward pass, the error signals propagate backward and the premise parameters are updated by gradient descent.

$$a_{ij}(t+1) = a_{ij}(t) - \frac{\eta}{p} \cdot \frac{\partial E}{\partial a_{ij}}$$

where η is the learning rate for a_{ij}. The chain rule is used to calculate the partial derivatives used to update the membership function parameters.

$$\frac{\partial E}{\partial a_{ij}} = \frac{\partial E}{\partial f} \cdot \frac{\partial f}{\partial f_i} \cdot \frac{\partial f_i}{\partial w_i} \cdot \frac{\partial w_i}{\partial \mu_{ij}} \cdot \frac{\partial \mu_{ij}}{\partial a_{ij}}$$

The partial derivatives are derived as

$$E = \frac{1}{2}(f - f^t)^2 \text{ hence } \frac{\partial E}{\partial f} = (f - f^t) = e$$

$$f = \sum_{i=1}^{n} f_i \text{ hence } \frac{\partial f}{\partial f_i} = 1$$

$$f_i = \frac{w_i}{\sum_{i=1}^{n} w_i}(p_i x + q_i y + r_i) \text{ hence,}$$

$$\frac{\partial f_i}{\partial w_i} = \frac{(p_i x + q_i y + r_i) - f}{\sum_{i=1}^{n} w_i}$$

$$w_i = \prod_{i=1}^{n} \mu_{A_{ij}} \text{ hence } \frac{\partial w_i}{\partial \mu_{ij}} = \frac{w_i}{\mu_{ij}}$$

The last partial derivative depends on the type of membership functions used. The parameters of the other membership functions are updated in the same fashion.

The gradient is then obtained as

$$\frac{\partial E}{\partial a_{ij}} = e\frac{(p_i x + q_i y + r_i) - f}{\sum_{i=1}^{n} w_i}\frac{w_i}{\mu_{A_{ij}}}\frac{\partial \mu_{A_{ij}}}{\partial a_{ij}}$$

$$\frac{\partial E}{\partial b_{ij}} = e\frac{(p_i x + q_i y + r_i) - f}{\sum_{i=1}^{n} w_i}\frac{w_i}{\mu_{B_{ij}}}\frac{\partial \mu_{B_{ij}}}{\partial b_{ij}}$$

The consequent parameters thus identified are optimal (in the consequent parameter space) under condition that the premise parameter are fixed. Acoordingly the hybrid approach is much faster than the strict gradient descent and it is worthwhile to look for the possibility of decomposing the parameter set in the maner of "$S = S_1 \oplus S_2$"

Roger Jang et al. [1] also showed that the computation complexibility of the least squares estimate is higher than that of the gradient descent. In fact, there are four methods to update the parameters, listed as below base on their computation complexities:

1) *Gradient Descent Only*: All parameters are updated by the gradient descent.
2) *Gradient Descent and One Pass of LSE*: The LSE is applied only once at the very begining to get the initial values of the consequent parameters and then the gradient descent takes over to update all parameters.
3) *Gradient Descent and LSE*: This is presented in detail by *Roger Jang et al.* in [1].
4) *Sequential (Approximate) LSE Only*: The ANFIS is linearized with respect to the premise parameters and the extended Kalman filter algorithm is employed to update all parameters.

In applications, the choice of these four methods above should be based on the trade-off between computation complexity and resulting performance.

The title of the paper is centered 17.8 mm (0.67") below the top of the page in 24 point font. Right below the title (separated by single line spacing) are the names of the authors. The font size for the authors is 11pt. Author affiliations shall be in 9 pt.

2.2. Mamdani ANFIS (M-ANFIS)

2.2.1. The network architecture

As given by Yuanyuan Chai et al. in [8, 9], a general M-ANFIS is described in shortly as:

Rule 1: If x is A_1 and y is B_1 then z_1 is C_1
Rule 2: If x is A_2 and y is B_2 then z_2 is C_2

The general model is displayed as *Figure 2*

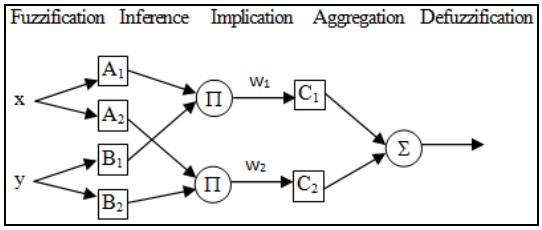

Figure 2: General model of Mamdani ANFIS

Layer 1: Fuzzification layer

$O_{1,i} = \mu_{A_i}(x)$, *i=1, 2*; $O_{1,i} = \mu_{B_{i-2}}(x)$, *i=3,4.*

the membership function is the generalized bell function

$$\mu_{A_i}(x) = \exp\left\{-\frac{1}{2}\left(\frac{x - c_i}{\sigma_i}\right)^2\right\}, \text{ or}$$

$$\mu_{A_i}(x) = \frac{1}{1 + \left[\left(\frac{x - c_i}{d_i}\right)^2\right]^{b_i}}$$

where $\{c_i, \sigma_i\}$ (or $\{b_i, c_i, d_i\}$) is the parameters set referred to as premise parameters.

Layer 2: Inference layer or Rule layer

$O_{2,i} = \omega_i = \mu_{A_i}(x) \times \mu_{B_i}$, i=1,2. (The firing strength ω_i is generated with product method)

Layer 3: Implication layer

$O_{3,i} = \omega_i \circ C_i;\ i = 1, 2.$

(Implication operator is product)

Layer 4: Aggregation layer

$O = \sum \omega_i \circ C_i;\ i = 1, 2$

Aggregation operator is sum. The consequent parameters are determined by C_i. If the consequent membership function (MF) is trapezoidal MF, each MF has 4 nonlinear parameters to be adjusted.

Layer 5: Defuzzification layer

$$O_5 = f = D \circ O_4$$

The crisp output f is archieved with the defuzzification method, COA (center of Area).

2.2.2. Learning procedures

Learning procedure presented by Yuanyuan Chai et al. [8] was applied the Gradient Descent method for all model parameters modifications and all these parameters are nonlinear parameters. When there is adequate training data, we can achieve M-ANFIS model.

Weight updating formulas are very important for adjusting M-ANFIS model parameters. This can be conclude as follow:

The task here is to minimize an overall error measure defined as:

$E_p = \sum_{k=1}^{N(L)} (d_k - x_{L,k})^2$; d_k is the k^{th} component of the p^{th} desired output vector; $x_{L,k}$ is the k^{th} component of the predicted output vector produced by presenting the p^{th} input vector to the network.

In order to minimize the overall error measure, we have to calculate the gradient vector, which is defined as the derivative of the error measure w.r.t. the parameter variables.

In order to calculate the gradient vector, error signal must be obtained, which is defined as the derivative of the error measure w.r.t. the output of a neuron. Once we obtain the gradient vector by chain rules, we can conclude the parameters updating formula for the whole network. The main idea is to go along (learn) against the direction of the gradient vector, and update the parameters by learning rules, eventually we can minimize the overall error measure for network output.

The weight updating formula in M-ANFIS

$$\Delta\omega_{ij} = -\eta\frac{\partial E}{\partial \omega_{ij}} = -\eta\frac{\partial E}{\partial x_i}\frac{\partial f_i}{\partial \omega_{ij}} = -\eta\varepsilon_i\frac{\partial f_i}{\partial \omega_{ij}} \quad ;$$

$$\omega_{next} = \omega_{now} - \eta\frac{\partial E}{\partial \omega_{ij}}$$

Where: j<i, that is, $x_i = f_i(\sum\omega_{ij}x_j + \theta)$ in which: f_i is the activation function of node i; x_i is the output of node i.

The error signals ε_i are propagated from the output layer back to input layer, layer by layer. And, the error signals of each node can be derived by error signals in previous layer nodes.

With assuming that $\omega_{ij} = \frac{\partial f_i}{\partial x_j}$ (or $x_j = \frac{\partial f_i}{\partial \omega_{ij}}$), the weight updating formula is rewritten as $\Delta\omega_{ij} = \eta\varepsilon_i x_j$, the parameter updating formula for each node can be derived, and the weights in whole network can be updated.

The general weight-updating formula is:

$$\Delta\omega_{ij} = -\eta(d_i - x_i)x_j\mathbf{X}$$

where: η is learning step (learning rate), d_i is the desired output for node i, x_i is the real output for node i, x_j is the input for node i, $\mathbf{X}$ is a Polynomial (usually, $\mathbf{X}$ is $x_i \times (1 - x_i)$).

3. ANFIS-alike for training a Fuzzy Constraint network

For simplicity, assume that given a fuzzy constraint network is displayed in Figure 3 as follow:

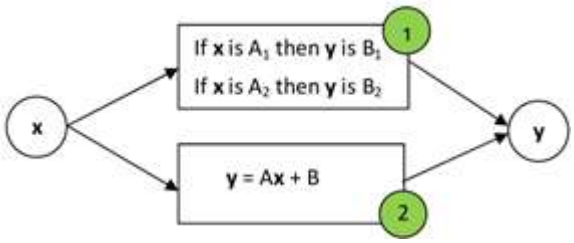

Figure 3: A fuzzy constraint network

wherein: A, A_1, A_2, B, B_1, B_2 are the fuzzy sets.

We can consider that, this network has two constraints: Constraint (1) is the set of two fuzzy rules; and, Constraint (2) is a fuzzy equation.

In this section, we described how to design a ANFIS-alike for training a fuzzy constraint network whose architecture was called as *C-ANFIS*. Besides, we also introduced its learning procedures and some experiments results.

3.1. The network architecture

The C-ANFIS architecture is designed by applying the Algorithm 1. "Designing the C-ANFIS architecture", step by step.

***Algorithm 1**. Designing the C-ANFIS architecture*

The algorithm includes 4 steps descibing as followings:

1) Deriving from Constraint (1) (set of the rules), design the architecture as the same as ANFIS
2) Add the "Weakly satisfy" layer to the architecture
3) Add the "Strongly satisfy" layer to the network architecture which is obtained in Step 1
4) Finishing and completing the finally network architecture.

Applying *Algorithm 1*, the network architect obtained after each step is descibed in detail as Figure 4.

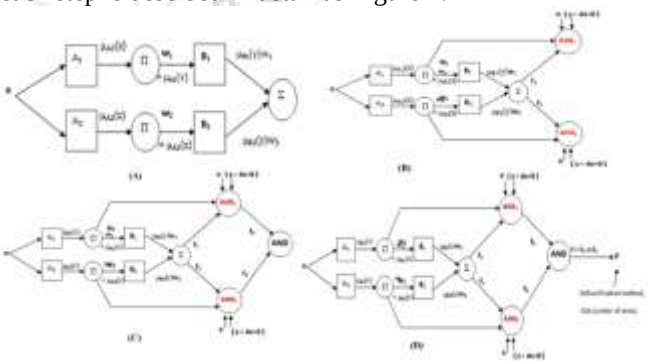

Figure 4: Network architecture obtained after applying Algorithm 1

(A): Network architecture after applying step 1;
(B): Network architecture after applying step 2
(C): Network architecture after applying step 3;
(D): Network architecture after applying step 4.

**** The C-ANFIS architecture***

As a result, the network architecture, which is obtained at the last step of applying *Algorithm 1*, is our proposed network architecture called as C-ANFIS.

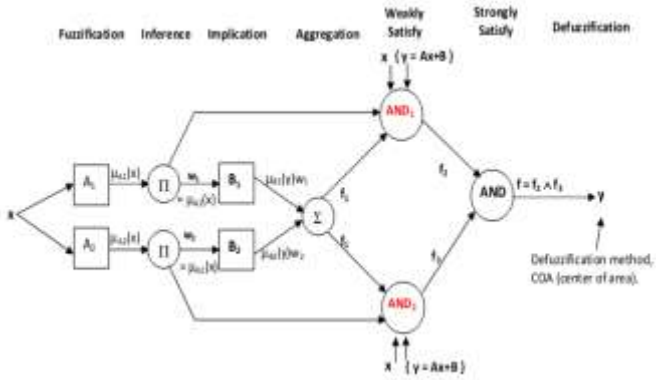

Figure 5: The C-ANFIS architecture

3.2. Outputs of the nodes in the network

Straightforwardly, the node's outputs of the proposed network can be derived from M-ANFIS, as below:

Layer 1: Fuzzification layer

$$O_{1,i} = \mu_{A_i}(x); \quad i = 1,2$$

The membership function is generalized function as:

$$\mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{d_i}\right)^2\right]^{b_i}};$$

where $\{a_i, b_i, c_i\}$ is the parameter set referred to as premise parameters.

Layer 2: Inference layer

$O_{2,i} = \omega_i = \mu_{A_i}(x)$, *i=1,2.*

Layer 3: Implication layer.

$O_{3,i} = \omega_i \circ B_i = \mu_{A_i}(x)\mu_{B_i}(y)$, *i=1, 2.*

(The implication operator is product)

Layer 4: Aggregation layer

$$O_4 = f_1 = \sum_i \omega_i \circ B_i$$
$$= \mu_{A_1}(x)\mu_{B_1}(y) + \mu_{A_2}(x)\mu_{B_2}(y), \; i=1, 2.$$

Layer 5: "Weakly satisfy" layer

$$O_{5,1} = f_2 = \sum_{i=1}^{2}\{\wedge(B_i, AA_1 + B)\}\,\omega_i$$
$$= \sum_{i=1}^{2}\left\{\mathbf{T}\left[\mu_{B_i}(y), \mathbf{S}\left(\mu_A(x)\mu_{A_1}(x), \mu_B(y)\right)\right]\right\}\omega_i$$
$$O_{5,2} = f_3 = \sum_{i=1}^{2}\{\wedge(B_i, AA_1 + B)\}\,\omega_i$$
$$= \sum_{i=1}^{2}\left\{\mathbf{T}\left[\mu_{B_i}(y), \mathbf{S}\left(\mu_A(x)\mu_{A_2}(x), \mu_B(y)\right)\right]\right\}\omega_i$$

Layer 6: "Strongly satisfy" layer

$$O_6 = \mathbf{T}(O_{5,1}, O_{5,2}) = \mathbf{T}(f_2, f_3)$$

where: **T** is T-Norm operator, **S** is S-Norm (or **T**-Conorm) operator. **T, S** can be one of the four operators below:

Minimum:	$T_{\min}(a,b) = \min(a,b) = a \wedge b$
Algebraic product:	$T_{ap}(a,b) = ab$
Bounded product:	$T_{bp}(a,b) = 0 \vee (a+b-1)$
Drastic product:	$T_{dp}(a,b) = \begin{cases} a, & if\ b=1 \\ b, & if\ a=1 \\ 0, & if\ a,b<1 \end{cases}$
Maximum:	$S_{\max}(a,b) = \max(a,b) = a \vee b$
Algebraic sum:	$S_{as}(a,b) = a+b-ab$
Bounded sum:	$S_{bs}(a,b) = 1 \wedge (a+b)$
Drastic product:	$S_{ds}(a,b) = \begin{cases} a, & if\ b=0 \\ b, & if\ a=0 \\ 1, & if\ a,b>0 \end{cases}$

(It is more intuitive if T_{min}, T_{ap}, S_{max}, S_{as} are selected.)

Layer 7: Defuzzification layer

$$O_7 = y = D \circ f$$

(Where: $\circ$ is composition operator. It can be max-min composition or max-product composition. This operator is almost the same as matrix multiplication).

The crisp output y is achieved with the defuzzification method, *COA* (Center of Area).

3.3. Learning procedures

It is simple to see that this network architecture is as same as the M-ANFIS architecture which is given by Yuanyuan Chai et al. **Error! Reference source not found.**, hence, the learning procedure for this network architecture can be applied as the same as the one in **Error! Reference source not found.**.

The task is minimize an overall error measure defined as:

$$E_p = \sum_{k=1}^{N(L)} (d_k - x_{L,k})^2$$

d_k is the k^{th} component of the p^{th} desired output vector; $x_{L,k}$ is the k^{th} component of the predicted output vector produced by presenting the p^{th} input vector to the network.

The weight updating formula in the network

$$\Delta\omega_{ij} = -\eta\frac{\partial E}{\partial \omega_{ij}} = -\eta\frac{\partial E}{\partial x_i}\frac{\partial f_i}{\partial \omega_{ij}} = -\eta\varepsilon_i\frac{\partial f_i}{\partial \omega_{ij}};$$
$$\omega_{next} = \omega_{now} - \eta\frac{\partial E}{\partial \omega_{ij}}$$

where: j<i, that is, $x_i = f_i(\sum\omega_{ij}x_j + \theta)$, in which: f_i is the activation function of node *i* and x_i is the output of node *i*.

The error signals ε_i are propagated from the output layer (Defuzzification layer) back to "Strongly satisfy" layer, then propagated back to "Weakly satisfy" layer, then propagated back to "Aggregation" layer, and then, this propagation is propagated continuously till to input layer.

With assuming that $\omega_{ij} = \frac{\partial f_i}{\partial x_j}$ (*or* $x_j = \frac{\partial f_i}{\partial \omega_{ij}}$), the weight updating formula, is rewritten as $\Delta\omega_{ij} = -\eta\varepsilon_i x_j$, the

parameter updating formula for each node can be derived, and the weights in whole network can be updated.

The general weight-updating formula is:

$$\Delta\omega_{ij} = -\eta(d_i - x_i)x_j X$$

where: η is learning step (learning rate), d_i is the desired output for node i, x_i is the real output for node i, x_j is the input for node i, X is a Polynomial (usually, X is $x_i \times (1 - x_i)$).

For the network architecture given above, this general weight-updating formula can be used for every layers in the network (including satisfy layers).

4. Conclusions

In this paper, a method for training a fuzzy constraint network is proposed. The so called C-ANFIS (Constraint-ANFIS) was proposed, also. By applying the algorithm named "Designing the C-ANFIS", we can obtain the C-ANFIS architecture deriving from ANFIS or M-ANFIS. We presented the way how to calculate the node's outputs of the network. Finally, a learning procedure was presented base on the basics of Hybrid learning procedures.

References

[1] Jyh-Shing Roger Jang, *ANFIS: Adaptive-Network-Based Fuzzy Inference System*, IEEE Transactions on System, Man, and Cybernetics, Vol. 23, No. 3, May/June 1993.
[2] Jyh-Shing Roger Jang, dept. of Computer Science, National Tsing Hua University Hsinchu, Taiwan, *Input Selection for ANFIS Learning*.
[3] Jyh-Shing Roger Jang and Chuen-Tsai Sun, *Neuro-Fuzy Modeling and Control*, IEEE, 1995.
[4] Ching-Yu Tyan, Paul P.Wang, Dennis R.Bahler, and Sathya P.Rangaswamy, *A New Methodology of Fuzzy Constraint-Based Controller Design via Constraint Network Processing*.
[5] James Bowen, Robert Lai, and Dennis Bahler, Dept. of Computer Science, North Carolina State University, USA, *Fuzzy Semantics and Fuzzy Constraint Networks*, IEEE, 1992.
[6] K. Robert Lai and Yi-Yuan Chiang, Dept. of Computer Science, Yuan Ze University, Taiwan, A *Constraint-based Framework for Incorporating A Priori Knowledge into Fuzzy Modelling*, IEEE, 2008.
[7] D. Dubois, H. Prade, and L. Ughetto, *A New Perspective on Reasoning with Fuzzy Rules*, Springer-Verlag Berlin Heidelberg 2002.
[8] Yuanyuan Chai, Limin Jia, and Zundong Zhang, *Mamdani Model based Adaptive Neural Fuzzy Inference System and its Applications*, World Academic of Science, Engineering and Techonology 51 2009.
[9] Yuanyuan Chai, Limiin Jia, Zundong Zhang, *Mamdani Model based Adaptive Neural Fuzzy In-ference System and its Application in Traffic Level of Service Evaluation*, 2009 sixth International Conference on Fuzzy Systems and Knowledge Discovery.
[10] Bryan Davis, University of Florida, System modeling using a Mamdani rule base.
[11] Deepak R.Keshwani, David D.Jones, George E.Meyer, and Rhonda M.Brand, Rule-based Madani-type fuzzy modeling of skin permeability, Elsevier, 2007.
[12] Robert Babuska and Henk Verbruggen, Delft University of Technology, Netherlands, Neuro-fuzzy methods for nonlinear system identification, Annual Reviews in Control 27, Elsevier, 2003.
[13] Oliver Elles, Martin Fischer, and Bernd Muller, Institute of Automatic Control Laboratory of Con-trol Engineering and Process Automation, Germany, Fuzzy Rule Extraction by a Genetic Algorithm and Constrained Nonlinear Optimization of Membership Functions, IEEE, 1996.
[14] David Lesaint, Intelligent System Lab, BTexact Technologies, BT France, France, Inferring Con-straint Types in Constraint Programming, Springer-Verlag Berlin Heidelberg 2002, CP 2002, LNCS 2470, pp. 492-507, 2002.
[15] G. Castellano, C. Castiello, A.M. Fanelli, and C. Mencar Dept. of Computer Science, University of Bari, Italy, Knowledge discovery by a neuro-fuzzy modeling framework, Elsevier, 2004.
[16] Babak Rezaee, M.H. Fazel Zarandi, Dept. of Industrial Engineering, Amirkabir University of Tech-nology, Iran, Data-driven fuzzy modeling for Takagi- Sugeno- Kang system, Elsevier, 2009.

Author Profile

Huy-Khoi Do was born in Vietnam. He obtained his master degree in Military Technical Academy. His research interests include fuzzy constraint network, communication sytems, images and speech processing, telecommunication.

Thi-Xuan Tran was born in Vietnam. She obtained his master degree in University of Information and Communication technology. Her research interests include computer science, fuzzy constraint network, data mining and machine learning.

Van-Nui Nguyen was born in Vietnam. He obtained his PhD degree in Department of Computer Science & Engineering from Yuan Ze University, Taiwan. His research interests include computer science, fuzzy constraint network, bioinformatics, computational proteomics and data mining.