Proceedings of 1993 International Joint Conference on Neural Networks

FEN (Fuzzy Expert Networks) learning architecture

Generalization of self adjusting Fuzzy modeling on event -driven acyclic neural networks using Expert Networks back propagation learning

Kazunari NARITA
Daido Steel Co.,Ltd.
add. ; Naka - ku, Nagoya 460 JAPAN
home add. ; 1-646, Higashi - Chaya,
Minato - ku, Nagoya 455 JAPAN

R.C.Lacher
Florida State University
Department of Computer Science
add. ; Tallahassee, FL 32306 USA

1 ABSTRACT

FEN (Fuzzy Expert Networks) is a new network architecture of neural objects for fuzzy modeling. The neural objects process information through node functions that are different from a typical sigmoidal node processor for an analog perceptron.
By connecting a few types of node processors on an event -driven acyclic (feed forward) neural network, FEN represents the fuzzy modeling with self adjustment. Weights on this network imply fuzzy parameters to be adjusted with no restrict of layered topology by learning.
FEN offers automated tuning from input-output data for membership functions on which the performance of fuzzy modeling depends. And especially using the enhanced idea of a dynamic backward error assignment for learning, FEN is effective for tuning parameters for non smooth membership functions, for example, symmetric triangular functions of an antecedent part.
Results of testing FEN are presented to demonstrate learning performance and adaptability.

2 INTRODUCTION

Fuzzy modeling, which has been successfully applied in both household and industry, has a high performance in describing and in function approximation for not only non-linear but also linear systems. [1-3]
The main characteristic of fuzzy modeling or fuzzy systems is embedded in linguistic control. Although it helps in describing and setting of initial value of parameters for membership functions using input from human operators, there still remains the problem of subtle tuning or adjusting of parameters from input-output data. [4,5]
Learning, where the main characteristic of artificial neural networks is embedded, implies correcting weights properly from input-output data. And back propagation learning for an artificial neural network was successfully popularized by PDP group. [6] This back propagation learning has a rich history of theory for a typical neural network i.e.,layered feed forward network with an analog perceptron.
Combining these two types of characteristics in fuzzy and neural networks is current research interest. [7 et.al]
This paper realizes a kind of fuzzy modeling on an event -driven acyclic (feed forward) neural network whose weights correspond to fuzzy parameters. But in order to utilize a weight correction step for this neural network using back propagation learning, theoretical foundation for learning is necessary, since this network is different from the typical one with the analog perceptron.
On the other side Lacher et al. introduced a method to represent an expert system as an event-driven acyclicneural network with a non analog perceptron called 'Expert Network'. Certainty factors of rule based reasoning are represented as weights in the networks. A generalized back propagation learning called ENBP (Expert Network Back Propagation) is used to adjust these weights. [8,9]
Not only for the knowledge extraction task of an expert system but also in the aspect of topology, function dynamism and node error assignment in back propagation learning for weight correction, their work that contributed to make it clear is summarized as follows.

1) With no restricts of layered topology, a generalized back propagation is valid under any node functions whose derivatives are well known on event-driven acyclic networks. Where backward error is propagated in the data flow fashion even if topology looks like the layered one.
2) The enhanced idea of a dynamically assigned influence factor as a refined method of backward error assignment is effective for generalized back propagation learning especially for the non-smooth node functions.

These can be applied to utilize a weight correction step of neural network for such an enlarged domain with analog input-output data as a high-level function or modeling, for example, fuzzy modeling. Thus we can generalize the new network architecture for self adjusting fuzzy modeling called 'FEN (Fuzzy Expert Networks) '.
As an example this paper demonstrates fuzzy modeling with non-smooth but widely used symmetric triangular membership functions of antecedent parts and with singleton of consequence on event-driven acyclic neural network topology. Where weights imply fuzzy parameter for membership functions of both antecedent and consequent part to be adjusted.
Results of the test example demonstrates learning performance and adaptability of 'FEN '.

3 FUZZY MODELING

This paper realizes a widely used simplified fuzzy inference modeling whose consequences are described with singletons and whose membership functions of antecedent parts have a symmetric triangular shape as an example. 【 3 et.al 】 The symmetric triangular function is popular, since it is faster to calculate and easy for the operator to understand.This model is expressed by the following.

3.1 RULE

$Rule\ i: If\ x_1\ is\ A_{i1}\ and\ ,...\ and\ x_m\ is\ A_{im}\ THEN\ y\ is\ c_i\ (i = 1,...n)$ (1)

Where i is the i-th fuzzy rule. $A_{i1},...A_{im}$ are membership functions of the antecedent part, n is the number of fuzzy rules, c_i is a singleton (real valiable).

3.2 MEMBERSHIP FUNCTIONS of the ANTECEDENT PART

The membership function A_{ij}, which has an example symmetric triangular shape, of the antecedent part in this case can be expressed as the product of linear threshold function $A1_{ij}$ and $A2_{ij}$.

$$A1_{ij}(x_j) = (((a_{ij}(x_j - b_{ij}) + 1) \vee 0) \wedge 1) \quad (2) \qquad A2_{ij}(x_j) = (((-a_{ij}(x_j - b_{ij}) + 1) \vee 0) \wedge 1) \quad (3)$$

$$A_{ij}(x_j) = A1_{ij}(x_j)\, A2_{ij}(x_j) \quad (i = 1,...,n) \quad (4)$$

Where a_{ij} is the gradient, b_{ij} is the center of symmetric triangular shape to be adjusted here. (see Fig. 1 linear threshold functions and triangular functions)

3.3 DEFUZZICATION

The defuzzication value (output) z can be derived as follows.

$$u_i = A_{i1}(x_1)\, A_{i2}(x_2)...A_{im}(x_m) \quad (5) \qquad z = \sum_{i=1}^{n} u_i c_i / \sum_{i=1}^{n} u_i \quad (6)$$

where u_i is a membership value of the antecedent part, c_i is a singleton to be adjusted here.

4 NODE DEFINITIONS

We define the term n-input 'node' to be a processing element consisting of the following in event driven artificial neural networks.
1)Real-valued input signal $z_1,...,z_n$ along with real-valued connection strength (weight) $w_1,...w_n$ for these signals. we call $z = (z_1,...,z_n)$ the input vector , $w = (w_1,...,w_n)$ the weight vector and x which is the product of z and w, $x = (z_1w_1,...,z_nw_n)$ the new input vector.
2)A combining function Γ that determines an internal state $y = \Gamma(x)$ from the input and weight vectors.
3)An output function ψ that determines a single output value $z = \psi(y)$ from the internal state. Taking the combining function to be the weighted sum of inputs and the output function to be a sigmoidal squashing function about some threshold gives what we refer to herein as an analog perceptron.

5 NODE FUNCTIONS for FUZZY MODELING

According to 4 NODE DEFINITIONS above we define under node functions for fuzzy modeling in this case.

5.1 PRODUCT NODE

combining function $\Gamma_{prod}(x_1,...,x_n) = \prod_{i=1}^{n} x_i$ (7) output function $\psi_{prod}(y) = y$ (8)

5.2 SUM NODE

combining function $\Gamma_{sum}(x_1,...,x_n) = \sum_{i=1}^{n} x_i$ (9) output function $\psi_{sum}(y) = y$ (10)

5.3 LINEAR THRESHOLD NODE

combining function $\Gamma_{lin}(x_1,...,x_n) = \sum_{i=1}^{n} x_i + 1$ (11)

output function

$$\psi_{lin}(y) = 1 \text{ if } y \geqq 1$$
$$= y \text{ if } 0 < y \leqq 1 \quad (12)$$
$$= 0 \text{ otherwise}$$

5.4 WEIGHTED SUM GRAVITY NODE

combining function $\Gamma_{grav}(x_1,...,x_n) = \sum_{i=1}^{n} x_i / \sum_{i=1}^{n} z_i$ (13)

output function $\psi_{grav}(y) = y$ (14)

5.5 CONFIGURATION

Fig.3 shows an example of the membership function of the antecedent part (2,3,4) of fuzzy modeling on FEN. Square node denotes INPUT node, circle node denotes SUM node, double square node denotes PRODUCT node and circle square node denotes LINEAR THRESHOLD node.
Fig. 4 shows an example of the rule and defuzzification part (5,6) on FEN. Double square node denotes PRODUCT node and triple square node denotes WEIGHTED SUM GRAVITY node.

6 EXPERT NETWORK BACK PROPAGATION

As described above Lacher introduced a theorem that back propagation, which has been successfully applied to layered feed forward networks of analog perceptrons, is valid under much more general node function on event-driven acyclic network without regard to layered structure. [8,9] Therefore we can calculate backward propagated error at each node in the data flow fashion even if network topology looks like the layered one.

6 .1 INFLUENCE FACTOR

Lacher also introduced an enhanced idea of dynamically assigned influence factor as a refined method of backward error assignment which is effective for generalized back propagation learning. [9] According to this factor, let the influence factor be ε_{kj}, e_j be the error assigned to node j, we have e_j in the following. (see 4 NODE DEFINITIONS)
For output node, error is assigned by using

$$e_j = I_j - z_j \quad (15)$$

For non-output node, error is assigned in term of successor node using

$$\varepsilon_{kj} = \psi'_k(y_k)(\partial\Gamma_k(x_{k1},...x_{kn})/\partial x_{kj}) \quad (16) \qquad e_j = \Sigma_k \varepsilon_{kj} w_{kj} e_k \quad (17)$$

Here k is the successor node to node j, e_k is the error at node k, ψ'_k is evaluated at the current internal state of node k, $\partial\Gamma_k/\partial x_{kj}$ is evaluated at the current input vector for node k, and w_{kj} is the weight between node k,j and I_j is an ideal output at node j.

6 . 2 GRADIENT DESCENT

Let total square error $E = \Sigma_j e_j^2$. An ideal output I_j is *assumed* if j is an output node and *defined* if j is not, in any case yielding the relation $e_j = I_j - z_j$. Suppose that node j has been designated for training. Denote the vector of synaptic weights on connections incoming to j by $w_1 = (w_{j1},...,w_{jn})$. The gradient ∇E of E with respect to w_j is the vector of partials

$$\partial E/\partial w_{ji} = 2e_j(\partial e_j/\partial z_j)(\partial z_j/\partial y_j)(\partial y_j/\partial x_{ji})(\partial x_{ji}/\partial w_{ji})$$
$$= -2e_j \psi'_j(y_j)(\partial\Gamma_j(x_{j1},...x_{jn})/\partial x_{ji})z_i. \quad (18)$$

Thus a step (with "learning rate" η and "momentum" μ) in the direction of ∇E is given

$$\Delta w_{ji} = \eta e_j \psi'_j(y_j)(\partial\Gamma_j(x_{j1},...x_{jn})/\partial x_{ji})z_i + \Delta w_{ji}^{prev}. \quad (19)$$

6. 3 DISCUSSION

Note that in Fig. 3 at the SUM NODE whose output value is $a_{ij}(x_j - b_{ij})$, if we use the concept of 'blame', which is used for an ordinary analog perceptron, the error $e_j = \Sigma_k w_{kj} e_k$ (20) is always zero! So we can't adjust the Δw_{ji} in (19).
But if we use the influence factor, the error at this SUM NODE is not necessarily always zero, so long as one output value of LINEAR THRESHOLD NODES is between 0 and 1. (in this case, at these LINEAR THRESHOLD NODES, one of these derivatives $\psi'_k(y_k)(\partial\Gamma_k(x_{k1},...x_{kn})/\partial x_{kj})$ in (16) is 1 and another is zero, therefore the error $e_j = \Sigma_k \varepsilon_{kj} w_{kj} e_k$ (17) is not zero,see (11), (12), and Fig.1.)

7 SIMULATION EXAMPLE

7 .1 SYSTEM

$$f(x_1, x_2) = (0.5x_1 + x_2^2 + \alpha)^2 \qquad (21)$$

Non linear system (21) is taken up for a simulation example. It has two input x_1, x_2 within [1, −1] using random numbers, one output y^{tr} normalized within [0, 1] and parameter α. Twenty data are employed for identification. The average square error is derived by (22) in the following.

$$E = (1/20) \sum_{q=1}^{20} (y_q - y_q{}^{tr}) \qquad (22)$$

where q implies a number of data, $y_q{}^{tr}$ implies a training data.

7 .2 LEARNING RATE

For simulation we use fuzzy modeling of 16 inference rules. (4 membership functions of each antecedent part for each 2 input and 16 singletons of consequent parts) We test using a different combination of learning rate of (19) for weights to be adjusted. ("momentum" μ is 1)
type a) ...learning rate of parameters for consequent parts are 7.5, for antecedent part are 0.75.
type b) ... learning rate of parameters for consequent parts are 7.5, for antecedent parts are 0.075.
type c) ... learning rate of parameters for consequent parts are 7.5, for antecedent parts are 0.
type c) implies training parameters only for consequent parts.

7 .3 TEST 1, LEARNING CURVE of FEN

Fig. 5 shows the relation between the number of iteration and average square error, E in (22). (α in (21) = 2.0)

7 .4 TEST 2, ADAPTABILITY of FEN

FEN is tested in parameter changing system (21) where parameter α is changing in 0.1 step. Learning stops when average square error E in (22) is less than 0.000225 for the first step. On this network the next learning starts until E in (22) is less than 0.000225 for next step.
Fig. 6 shows the relation between parameter α in (21) and the number of iteration for convergence.

7 .5 RESULT

Note that both in TEST 1 and TEST 2, among the 3 types of learning rate combination, type a) and type b) show better results than type c). This implies that compared with training parameter only for consequent part, training both for antecedent and consequent part is better for both identification and adaptation. This shows the usefulness of self adjusting parameters for both antecedent and consequent part of fuzzy modeling by this method.

8 CONCLUTIONS

FEN (Fuzzy Expert Networks) not only facilitates tuning parameters for membership functions of both antecedent and consequent part of fuzzy modeling but also has well adaptability.
Test result lead us to believe that event-driven neural computing with weight correction such as this method provides us a new generalized tool for parameter adjustment also in further high-level function or modeling for analog input-output data, for example, a more complicated fuzzy system. As a future work such an enhanced faster learning algorithm for generalized neural network as in vanilla back propagation is expected. [10 et.al]

9 REFERENCES

[1] M.Sugeno, Ed., *Industrial Applications of Fuzzy Control*, Amsterdam:North-Holland, 1985.
[2] M.Sugeno, IEEE Trans. Syst., Man, Cybern., Vol.15, no.1, p.116 – 132, 1985
[3] M.Mizumoto, "Realization of PID Controls by Fuzzy Control Methods" in Proc. of *IEEE International Conference on Fuzzy Systems* (San Diego), p.709 – 715, 1992.
[4] L.A.Zadeh, "Fuzzy logic" *IEEE Computer*, Apr.1988.
[5] E.H.Mamdani, in the Linguistic Synthesis of Fuzzy Controller, Int. J. Man-Machine Studies, Vol.8, No.6,669/678, 1976
[6] D.E.Rumelhart and J.L.McClelland, *Parallel Distributed Processing*,Cambridge,MA:MIT Press.
[7] B.Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ, 1992.
[8] R.C.Lacher, S.I.Hruska, D.C.Kuncicky, "Back Propagation Learning in Expert Networks" *IEEE-Transaction on Neural Networks*, Vol.3, No.1, Jan.1992.
[9] R.C.Lacher, "Node error assignment in expert networks" in *Intelligent Hybrid Systems*, A.Kandel and G.Langholz, Eds. Boca Ratan, FL:CRC Press.
[10] S.E.Fahlman "An Empirical Study of Learning Speed in Back-Propagation Networks" Tech. Rep. CMU-CS-88-162, Carnegie Mellon University, Pittsburg, PA, 1988

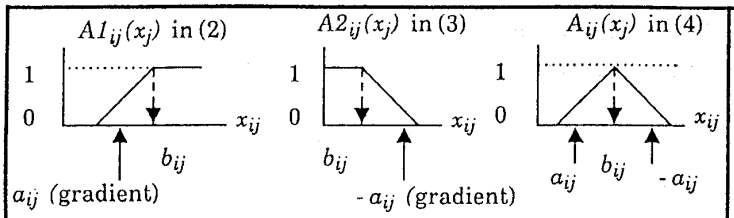

Fig.1 linear threshold functions and triangular functions

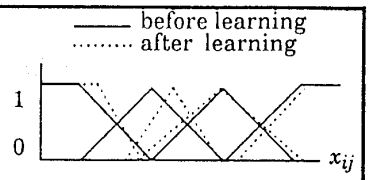

Fig. 2 membership functions of the antecedent part

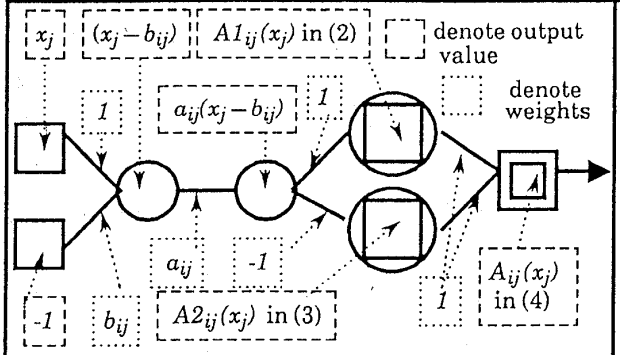

Fig.3 configuration of the antecedent part

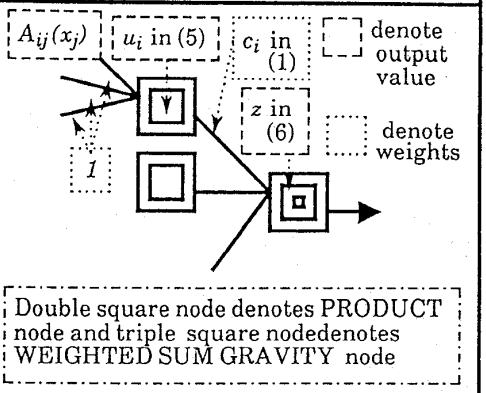

Fig. 4 configuration of the rule and consequent part

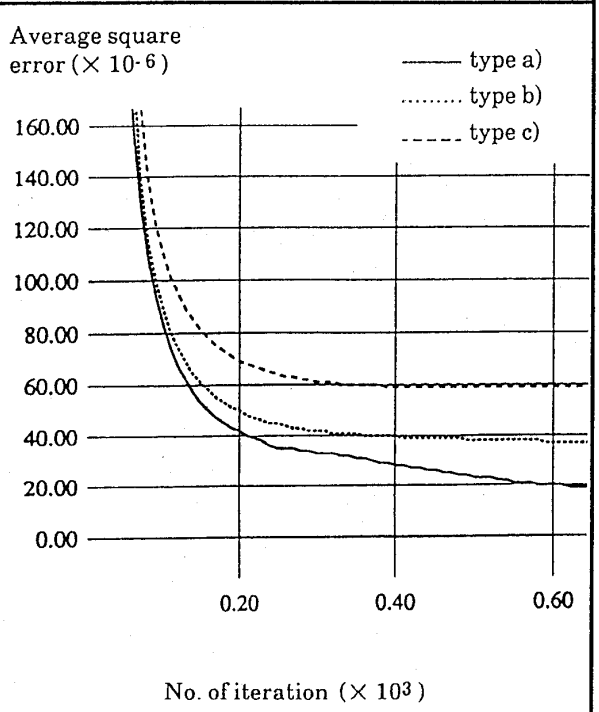

Fig. 5 learning curve

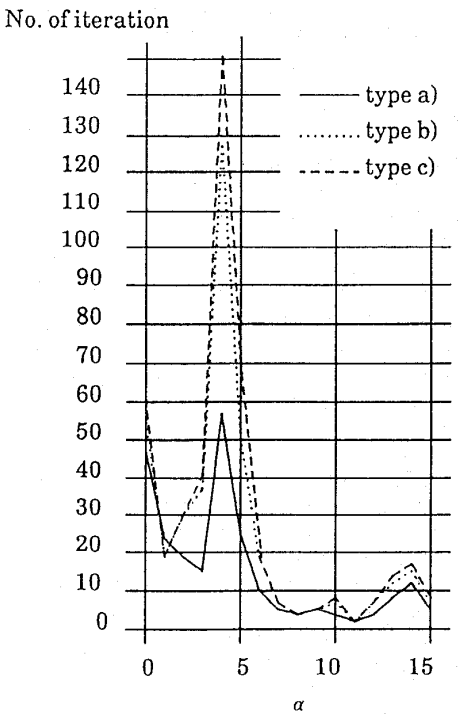

Fig. 6 adabtability