15th Triennial World Congress, Barcelona, Spain

A NEW NEUROFUZZY NETWORK FOR SELFORGANIZING CONTROL

Nicolae Constantin

Automatic Control and System Engineering Department
University "Politehnica" Bucharest
Spl. Independentei, nr.313, 77206, Bucharest, Romania.
Tel./Fax +40-1-4119918, E-mail: nicu@lnx.cib.pub.ro

Abstract: In this paper a novel neural fuzzy inference network (NFIN) it is proposed. The NFIN represent a modified Takagi-Sugeno-Kang (TSK) type fuzzy rule based model with neural network learning ability. The rules in the NFIN are created and adapted in an on-line learning algorithm. The structure learning together with the parameter learning forms the learning algorithms for the neural fuzzy network. It is proved that NFIN can greatly reduce the training time, avoid the overtuned phenomenon and has perfect regulation ability. *Copyright © 2002 IFAC*

Keywords: neural networks, fuzzy inference, on-line control, learning algorithms

1. INTRODUCTION

Neural fuzzy network approach become a popular research topic in these years. From the theoretical point of view, many effective learning algorithms for neurofuzzy systems were proposed. For example, Jang's (1993) adaptive-network-based fuzzy inference system, Lin's (1994) neural-network-based fuzzy logic control and decision system, Wang's (1994) several adaptive fuzzy systems, the fuzzy neural net with fuzzy inputs and fuzzy targets by Ishibuchi, *et al*. (1993), etc.

Neural fuzzy brings the low-level learning and computational power of neural networks into fuzzy systems, and provides the high-level human-like thinking and reasoning of fuzzy systems into neural networks. A recurrent neural network which involves dynamic elements in the form of feedback connections used as internal memories perform a dynamic mapping (Gilles, *et al*., 1994). In the recurrent neural fuzzy network proposed in this paper, the rules are constructed automatically during the on-line operation.

Two learning phases, the structure and the parameter learning are adapted on-line for the construction task. The clusters formed in the input space are aligned such that the number of rules and the number of membership functions are decreased. First the network is trained off-line to learn the inverse dynamics model of the plant , and then the network is configured as a feedforward network controller to the plant. Next, a conventional on-line training scheme is used to adapt the network to the practical environment.

This paper is organized as follows. Section 2 describes the architecture of the network. The on-line structure and parameter learning algorithm is presented in Section 3. The control results of using NFIN and BPNN on the water bath temperature are

presented in Section 4. Finally conclusions are summarized in the last section.

2. ARCHITECTURE OF THE NETWORK

It is a five-layered neural fuzzy network embedded with dynamic feedback connections. The function of each node in each layer is presented below.

Layer 1: Each node is called linguistic node and corresponds to one variable. The node only transmits input values to the next layer:

$$a^{(1)} = u_i^{(1)} \tag{1}$$

Layer 2: Each node is called input term node and corresponds to one linguistic label (small, large, etc.) of an input variable. A Gaussian membership function is used. The operation performed in this layer is:

$$a^{(2)} = \exp\left\{-\frac{\left(u_i^2 - m_{ij}\right)^2}{s_{ij}^2}\right\} \tag{2}$$

where m_{ij}, σ_{ij} are, respectively, the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i.

Layer 3: Nodes are called rule nodes and performs precondition matching of a rule. Each node has inputs from layer 2 that represents the rule's spatial firing degree and inputs from the feedback layer that represents the rule's temporal firing degree. The following AND operator is used on each rule node

$$a^{(3)} = a^{(6)} \prod_i u_i^{(3)} = a^{(6)} e^{-[D_i(x-m_i)]^T [D_i(x-m_i)]} \tag{3}$$

where D_i = diag $(1/\sigma_{i1}, \ldots, 1/\sigma_{in})$, m_i = $(m_{i1}, m_{i2}, \ldots, m_{in})^T$, and $a^{(6)}$ is the output of the feedback term node. The fuzzy AND operation used in (3) represent the algebraic product in fuzzy theory.

Layer 4: This layer is called the consequent layer and the nodes are called output term nodes. The functions of each output term node performs the following fuzzy OR operation:

$$a^{(4)} = \sum_i u_i^{(4)} \tag{4}$$

to integrate the fired rule which have the same consequent part.

Layer 5: Each node in this layer is called an output linguistic node and corresponds to one output linguistic variable. This layer performs the defuzzification operation.

The function performed in this layer is

$$y_j = a^{(5)} = \frac{\sum_i u_i^{(5)} \hat{m}_{ji}}{\sum_i u_i^{(5)}} \tag{5}$$

where $u_i^{(5)} = a^{(4)}$ and $\hat{m}_{ji}$, the link weight, is the center of the membership function of the ith term of the jth output linguistic variable.

Feedback layer: Two types of nodes are used in this layer. The square node named as context node is associated with a circle node named as feedback term node.

The context node functions as a defuzzifier

$$h_j = \sum_i a_i^{(4)} w_{ji} \tag{6}$$

where the internal variable h_j is interpreted as the inference result of the hidden (internal) rule and w_{ji} is the link weight from the ith node in layer 4 to the jth internal variable. The feedback term node evaluates the output by

$$a^{(6)} = \frac{1}{1+e^{-h_i}} \tag{7}$$

This output is connected to the rule nodes in layer 3, which connect to the same output term node in layer 4. The outputs of feedback term nodes contain the firing history of the fuzzy rules.

The following dynamic fuzzy reasoning is realised :
Rule i :
IF $x_1(t)$ is A_{i1} and … and $x_n(t)$ is A_{in} and $h_i(t)$ is G
THEN
$y_1(t+1)$ is B_{i1} and $y_2(t+1)$ is B_{i2} and
$h_1(t+1)$ is w_{1i} and … and $h_m(t+1)$ is w_{mi}

where x_i is the input variable, y_i is the output variable, A_{i1},…, A_{in}, G, B_{i1}, B_{i2} are fuzzy sets, h_i is the internal variable, w_{1I} and w_{mi} are fuzzy singletons, and n and m are the numbers of input and internal variables, respectively.
This fuzzy rule can be decomposed into two parts: the external rule and the internal rule. The external rule can be expressed by
Rule i :
IF $x_1(t)$ is A_{i1} and … and $x_y(t)$ is A_y and $h_{yy}(t)$ is G
THEN $y_1(t+1)$ is B_{i1} and $y_2(t+1)$ is B_{i2}
and the internal rule is
Rule i :
IF $x_1(t)$ is A_{i1} and … and $x_n(t)$ is A_{in} and $h_i(t)$ is G
THEN $h_1(t+1)$ is w_{1i} and … and $h_m(t+1)$ is w_{mi}
In time domain a rule is fired by the outputs of rules which were fired one time step ahead.

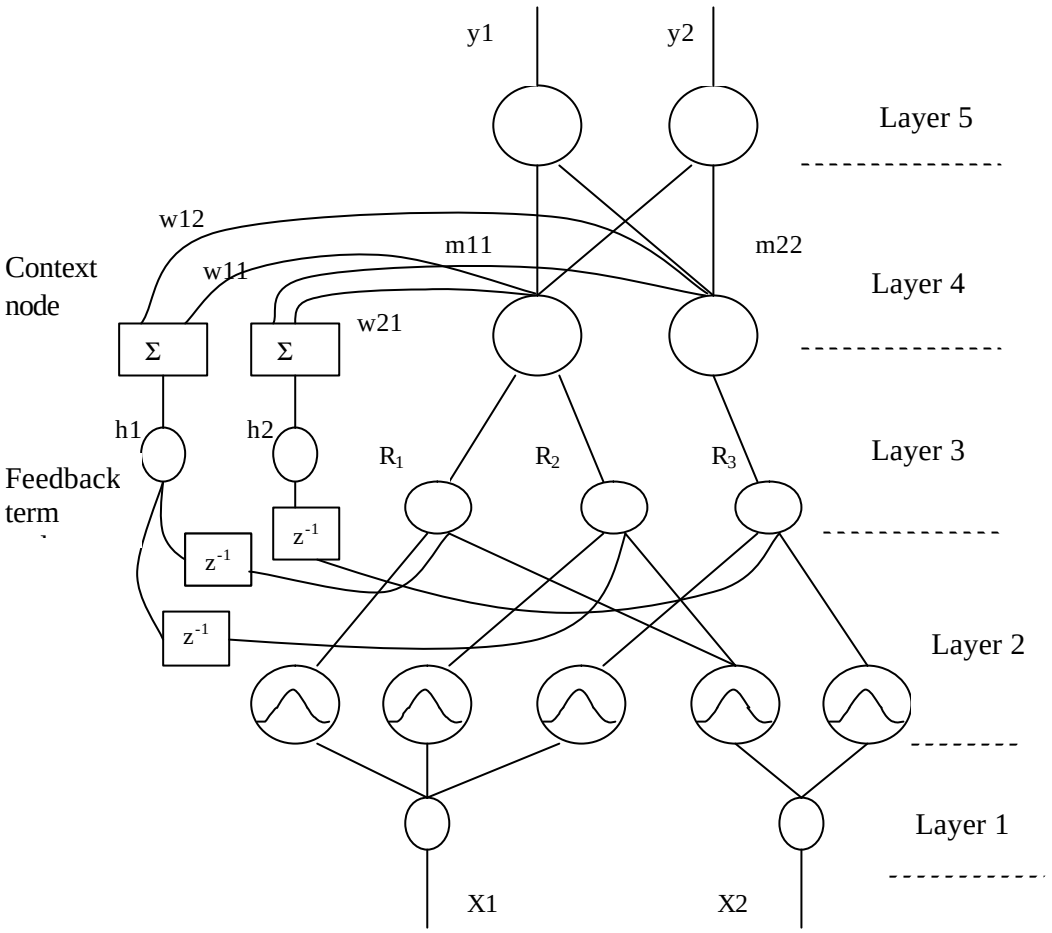

Fig. 1. Structure of the self-organizing neural fuzzy inference network

3. LEARNING ALGORITHMS

The structure and parameter learning are the two phases of learning that are used concurently in the network. The structure learning includes the precondition, consequents, and feedback structure identification of dynamic fuzzy if-then rules.

3.1 Input-output partitioning

The number of rules depends on the way the input space is partitioned. Here only the spatial information is used for clustering and the spatial firing strength is used as degree measure

$$F^i(x) = \prod_i u_i^{(3)} = e^{-[D_i(x-m_i)]^T[D_i(x-m_i)]} \qquad (8)$$

where $F^i \in [0,1]$. The exponent represent the distance between x and the center of cluster i. Based on this measure the criterion for the generation of a new fuzzy rule is as follows.

Let $\mu = (m_i, \sigma_i)$ represent the Gaussian membership function with center m_i and width σ_i, and E(A,B) represent the fuzzy measure of two fuzzy sets A and B. It is suppose that no rule exists initially. The algorithm for the generation of new fuzzy rules and fuzzy sets for each input variable is as follows.

IF x is the first incoming pattern THEN do
{ Generate a new rule, with center $m_1 = x$, width $D_1=\text{diag}(1/\sigma_0, \ldots, 1/\sigma_0)$, where σ_0 is a prespecified constant.
After decomposition, we have n one-dimension membership functions,
with $m_{1i}=x_i$, and $\sigma_{1i}=\sigma_0$, i =1,…, n.
}
ELSE
for each newly incoming pattern x, do
{
find J = argmax $_{1 \le j \le c(t)}$ $F^j(x)$,
IF $F^J \ge F_{in}(t)$ THEN
after o period of time perform fuzzy measure and eliminate unnecessary membership functions
ELSE
{ c(t+1) = c(t) + 1, generate a new fuzzy rule, with $m_{c(t+1)} = x$, $D_{c(t+1)} = (-1/\beta)\ \text{diag}\ (1/\ln(F^J), \ldots, 1/\ln(F^J))$.
After decomposition, we have $m_{new\text{-}i} = x_i$, $\sigma_{new\text{-}i} = -\beta \ln(F^J)$, i = 1,…,n.

Calculate the fuzzy measure for each input variable: {degree (i, t) = max $_{1 \le j \le ki}$ E [μ ($m_{new\text{-}i}$, $\sigma_{new\text{-}i}$), μ (m_{ji}, σ_{ji})] where k_i is the number of partitions of the ith input variable.
IF degree(i, t) $\le \rho$, THEN
adopt this new membership function and set $k_i = k_i + 1$,
ELSE
set the projected membership function as the closest one.
}
}
}

3.2. Fuzzy rules

The generation of a new cluster correspond to the generation of a new fuzzy rule with its preconditioned part constructed by the learning algorithm. When a new input cluster is formed after the presentation of the current input-output training pattern (x, d), the consequent part is constructed by the following algorithm:

IF there are no output clusters,
Do {part1 from the previous process, with x replaced by d}
ELSE
do{
find J = argmax $_j$ F^j(d);
IF $F^J \ge F_{out}(t)$ THEN
connect input cluster c(t+1) to the existing output cluster J
ELSE
generate a new output cluster connect input cluster
c(t+1) to the newly generated output cluster.
}
Since only the center of each output membership function is used for defuzzification, the consequent part of each rule may be simply regarded as singleton.

3.3. Feedback structure identification

During the on-line learning a context node is created once an output cluster is created. The inputs for the context nodes comes from all the neurons in layer 4 with the link weight assigned with a small random value in [-1,1]. Each internal variable has assigned a global membership function and acts as the feedback term node of the corresponding context node.

In general each rule node has its own corresponding internal variable, which is to memorize the firing history of the rule. Due to the feedback term node is connected to the rule that maps to the corresponding output cluster it memorizes the parameter number in the feedback layer can be effectively reduced.

3.4. Parameter identification

After the structure is adjusted according to the current training pattern, the network starts the parameter identification phase to adjust the parameters optimally.
The learning is performed on the whole network. In order to derive the learning algorithm the ordered derivative method (Sastry, *et. al,* 1994) is use. This is a partial derivative whose constant and varying terms are defined using an ordered set of equations.
The goal is to minimize the error function

$$E(t+1) = 1/2\left[y_j(t+1) - y_j^d(t+1)\right]^2 \qquad (9)$$

where $y_j^d(t+1)$ is the desired output and $y_j(t+1)$ is the current output.

For each training pattern, starting at the input nodes, a feedforward pass is used to compute the activity levels of all the nodes in the network to obtain the current output.
The update rules for the free parameters are as follows:
- update rule of $\hat{m}_{ji}$ (the center of the output membership function)

$$\hat{m}_{ji}(t+1) = \hat{m}_{ji}(t) - h\frac{\partial E}{\partial \hat{m}_{ji}(t)}(t+1) \qquad (10)$$

Based on the same strategy are update the others free parameters: m_{pq} (the center of the membership function in the precondition part) , σ_{pq} (the width of the membership function in the precondition part), w_{pq} (the memory weight parameter in the feedback layer). In (Sastry, *et. al*, 1994) it is suggested that learning constant η_w may be chosen several times larger than η such that one could obtain about the same convergence speed of all parameters.

4. SIMULATION RESULTS

Consider a discrete-time SISO temperature control system :

$$y_p(k+1) = a(T)y_p(k) + \frac{b(T)}{1+e^{1/2y_p(k)-g}}u(k) + [1-a(T)]y_0 \qquad (11)$$

where

$$a(T) = e^{-aT},$$
$$b(T) = \frac{a}{b}(1-e^{-aT}). \qquad (12)$$

This model is given in (Lin, *et al*., 2000) and represents the equation for a real water temperature control system. Another example can be found in (Dumitrache, *et al*., 2000).

The parameters are set as α = 1.00151 e-4 , β = 8.67973e-3, γ = 40 and y0 = 25°C. The plant input u(k) is limited between 0 and 5V and the sampling period is chosen T = 30s. The task is to control the water bath system to follow three set-points:

$$r(k) = \begin{cases} 35^\circ C, & k \le 40, \\ 55^\circ C, & 40 < k \le 80 \\ 75^\circ C, & 80 < k \le 120. \end{cases} \qquad (13)$$

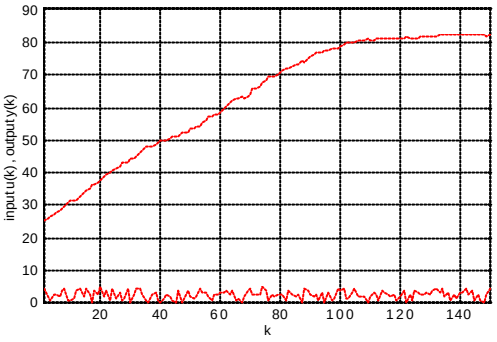

Fig. 2. Input –output characteristic of the process

Control performance

A sequence of random input signals u(k) is injected directly to the system described in equation (11). Then an open-loop input-output characteristic of the system is obtained as shown in fig.2. It is observed that the system behaves linearly up to about 70°C and then becomes nonlinear and saturates at about 82°C. From the input-output characteristic of the system 100 training patterns are selected to cover the entire reference output space.

For the BPNN, a four-layer feedforward network with two hidden layers is used. The hidden and output nodes have hyperbolic tangent sigmoid activation functions. The number of hidden nodes is usually decided by trials and errors.

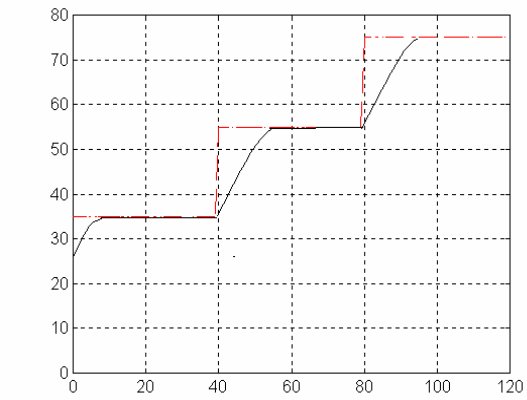

Fig. 3 Regulation performance of control system in the 20th trial of on-line training using BPNN(15)

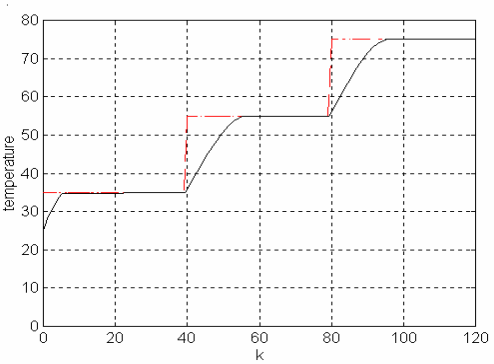

Fig. 4. Regulation performance of control system in the 20th trial of on-line training NFIN

For the NFIN the learning parameters are chosen as $\eta = \eta_w$, = 0.005, ρ = 0.9, F_{in} = 0.1, F_{out} = 0.7. Using the input vector [y_p(k), y_p(k+1] and target pattern u_d(k), after 10000 iterations the error for BPNN(15,15) only reach about 0.7231. For the NFIN the error reaches 0.6652 after only 6 iterations. However the errors are still large and on-line training is necessary to achieve high accuracy.

In the on-line training the NFN and BPNN are trained by the conventional on-line training scheme. To test the regulation performance in each trial, a performance index, sum of absolute error (SAE) is defined by :

$$SAE = \sum_{k=1}^{120} \left| r(k) - y_p(k) \right| \qquad (14)$$

where r(k) and y_p(k) are the reference output and the actual output of control system, respectively. The performance index is calculated for a trial with k ranging from 1 to 120. After 20th trial of on-line training the SAE = 345.31 for BPNN and 338.41 for NFN.

The number of generated rules is 7, and the number of fuzzy sets on the y(k) and y(k+1) dimensions are 4 and 4 respectively. NFN can reduce the training time and avoid the overtuned phenomenon. It has also good regulation control capability. The number of network structure parameters the NFN also use less parameters than the BPNN.

5. CONCLUSIONS

A fuzzy neural network with on-line self-organizing learning ability is proposed in this paper. It perform fuzzy reasoning by creating fuzzy rules, which are generated automatically and optimally during on-line operation via concurrent structure and parameter learning. The structure identification can reduce the rule number and network size. The final regulation performance of the NFN controller in the conventional on-line shows the best regulation control capability. The NFN overcome the disadvantages of BPNN and has good control capability.

REFERENCES

Dumitrache, I., N.Constantin, and I. Mihu (2000). Selforganizing Neural Fuzzy Inference System for Intelligent Control, *Proc. ISIC 2000*, Patras ,pp.211-215.

Gilles, C.L., G.M.Kuhn, and R.J.Williams (1994). Dynamic recurrent neural networks: Theory and applications, *IEEE Trans. Neural Networks*, vol. **5**, pp.153-156.

Ishibuchi, H. K. Kwon and H.Tanaka (1993). Learning of fuzzy neural networks from fuzzy inputs and fuzzy targets", *Proc. 5th IFSA World Congr.*,vol.I, pp. 147-150.

Jang, J.R. (1993). ANFIS: Adaptive-network-based fuzzy inference systems, *IEEE Trans.Syst.,Man, Cybern*., vol. **23**, pp. 665-685.

Lin, C.T., Juang C.F., Li C.P.(2000). Water bath temperature control with a neural fuzzy inference network, *Fuzzy Sets and Systems*, **11**, pp. 185-206

Sastry, P.S., G.Santharam, and K.P. Unnikrishnan, (1994). Memory neural networks for identification and control of dynamic systems, *IEEE Trans. Neural Networks*, vol. **5**, pp. 306-319.

Wang, L.-X. (1993). *Adaptive Fuzzy Systems and Control*. Englewood Cliffs, NJ: Prentice-Hall.