A Multilayer Feedforward Fuzzy Neural Network

Aydoğan Savran

Ege University, Department of Electrical and Electronics Engineering,
35100, İzmir, Turkey
aydogan.savran@ege.edu.tr

Abstract. This paper describes the architecture and learning procedure of a multilayer feedforward fuzzy neural network (FNN). The FNN is designed by replacing the sigmoid type activation function of the multilayer neural network (NN) with the fuzzy system (FS). The Levenberg-Marquardt (LM) optimization method with a trust region approach is adapted to train the FNN. Simulation results of a nonlinear system identification problem are given to show the validity of the approach.

1 Introduction

Multilayer feedforward neural network (NN), also referred as multilayer perceptron (MLP), is one of the most popular NN architecture. The MLP consist of weighted connections and neurons arranged in layers. Each neuron collects the values from all of its input connections and produces a single output passing through an activation function. The sigmoid type activation functions are usually used. One way to improve the NN performance is to modify the activation function. The modifications of the activation function are highly investigated in the literature [1],[2],[3],[4].

A different approach to define activation functions is to use the fuzzy logic methodology. A fuzzy system can transform a knowledge base into a nonlinear mapping. When a fuzzy system designed or trained, the IF-THEN rules which are easily interpreted can be obtained. Recently, Olivas et al. proposed to use a fuzzy based activation function to increase the interpretation and simplify the hardware implementation of the neurons [5]. Oysal et al. developed a dynamic fuzzy network consist of the fuzzy activation function to model dynamic systems [6].

In the present paper, we propose the use of the fuzzy system as the activation function of the MLP. We aim to combine the MLP computational power with the fuzzy system interpretability in the same scheme. Since the MLP and the fuzzy system we used are both feedforward and universal approximators, we can conclude that the resulting FNN also conveys these features. The FNN is trained with a similar procedure with the MLP. So the learning methods of the MLP can be modified for the proposed FNN. We adapted the Levenberg-Marquardt (LM) optimization method with a trust region approach which provides fast convergence to train the FNN [7].

F.A. Savacı (Ed.): TAINN 2005, LNAI 3949, pp. 78 – 83, 2006.

2 The FNN architecture

The architecture of the proposed FNN is depicted in Fig. 1. The FNN is a modified model of the MLP with the fuzzy system (FS) activation function. We chose the FS with product inference engine, singleton fuzzifier, center average defuzzifier, and Gaussian membership functions (MFs) are of the following form

$$f_j(x_j) = \frac{\sum_{r=1}^{R_j} \bar{y}_{jr} \exp(-\frac{1}{2}(\frac{x_j - \bar{x}_{jr}}{\sigma_{jr}})^2}{\sum_{r=1}^{R_j} \exp(-\frac{1}{2}(\frac{x_j - \bar{x}_{jr}}{\sigma_{jr}})^2} \tag{1}$$

where x_j is the input of the FS, R_j is the number of the rules in the fuzzy rule base, $\bar{y}_{jr}$, $\bar{x}_{jr}$, and σ_{jr} are the free fuzzy sets parameters [8]. $\bar{x}_{jr}$ and σ_{jr} are the center and the spread of the r^{th} rule of the j^{th} neuron, respectively. An output membership function which has a fixed spread of 1, can be characterized with only the center parameter ($\bar{y}_{jr}$).

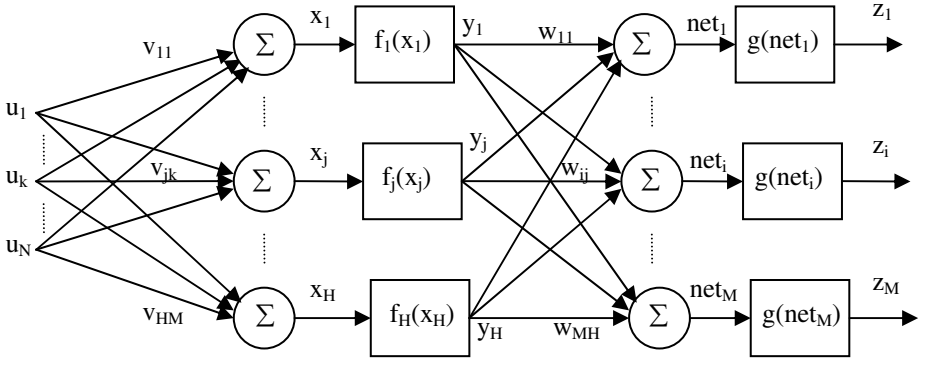

Fig. 1. The FNNN architecture

The output of the FNN computed as:

$$\begin{aligned} &x_j = \sum_k v_{jk} u_k, \quad y_j = f_j(x_j), \quad net_i = \sum_j w_{ij} f_j(x_j), \quad z_i = g_i(net_i) \\ &z_i = g_i(\sum_j w_{ij} f_j(\sum_k v_{jk} u_k)) \\ &k = 1,...,N \qquad j = 1,...,H \qquad i = 1,...,M \end{aligned} \tag{2}$$

where f(.) and g(.) are the hidden layer and the output layer activation functions respectively. In the hidden layer, the activation function is the fuzzy system. In the output layer, fuzzy, sigmoid or linear type activation functions can be used.

3 Learning

The output sum squared error can be defined as the cost function

$$E(p) = \frac{1}{2}\sum_i (z_i - d_i)^2 = \frac{1}{2}\sum_i e_i^2 \tag{3}$$

where z is the FNN output, d is the desired output, and p represents the free parameters of the FNN (v_{jk}, w_{ij}, $\bar{y}_{jr}$, $\bar{x}_{jr}$ and σ_{jr}). In order to improve the convergence rate and accuracy of the steepest descent algorithm, we adapted the Levenberg-Marquardt algorithm with a trust region approach to train the FNN. The LM method requires the Jacobian matrix J that contains the partial derivatives of the error terms with respect to free parameters of the FNN as

$$J = \begin{bmatrix} \frac{\partial e_1}{\partial w_{ij}} & \frac{\partial e_1}{\partial v_{jk}} & \frac{\partial e_1}{\partial \bar{x}_{jr}} & \frac{\partial e_1}{\partial \sigma_{jr}} & \frac{\partial e_1}{\partial \bar{y}_{jr}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial e_M}{\partial w_{ij}} & \frac{\partial e_M}{\partial v_{jk}} & \frac{\partial e_M}{\partial \bar{x}_{jr}} & \frac{\partial e_M}{\partial \sigma_{jr}} & \frac{\partial e_M}{\partial \bar{y}_{jr}} \end{bmatrix} \tag{4}$$

where ($\bar{y}_{jr}$, $\bar{x}_{jr}$ and σ_{jr}) represent all of the parameters of the fuzzy systems and (v_{jk} and w_{ij}) represent the weights in the hidden and output layers. The sensitivity of the error with respect to weights of the output layer can be computed as

$$\begin{aligned} &\frac{\partial e_i}{\partial w_{ij}} = \frac{\partial e_i}{\partial z_i}\frac{\partial z_i}{\partial net_i}\frac{\partial net_i}{\partial w_{ij}} \\ &\text{where} \quad \frac{\partial e_i}{\partial z_i} = 1, \quad \frac{\partial z_i}{\partial net_i} = g_i'(net_i) = \delta_i, \quad \frac{\partial net_i}{\partial w_{ij}} = z_j \\ &\frac{\partial e_i}{\partial w_{ij}} = g_i'(net_i) y_j = \delta_i y_j \end{aligned} \tag{5}$$

Then, we can carry out the similar process at the hidden layer.

$$\begin{aligned} &\frac{\partial e_i}{\partial v_{ij}} = \frac{\partial e_i}{\partial z_i}\frac{\partial z_i}{\partial net_i}\frac{\partial net_i}{\partial y_j}\frac{\partial y_j}{\partial x_j}\frac{\partial x_j}{\partial v_{jk}} \\ &\text{where} \quad \frac{\partial net_i}{\partial z_j} = w_{ij}, \quad \frac{\partial y_j}{\partial x_j} = f_j'(x_j), \quad \frac{\partial x_j}{\partial v_{jk}} = u_k \\ &\frac{\partial e_i}{\partial v_{jk}} = (f_j'(x_j)\sum_i \delta_i w_{ij}) u_k \end{aligned} \tag{6}$$

where $f'(x)$ is the input-output sensitivity of the fuzzy system. It is computed as

$$f'_j(x_j)=\sum_{r=1}^{R_j}\frac{y_j-\bar{y}_{jr}}{\sum_{r=1}^{R_j}\exp(-\frac{1}{2}(\frac{x_j-\bar{x}_{jr}}{\sigma_{jr}})^2)}\exp(-\frac{1}{2}(\frac{x_j-\bar{x}_{jr}}{\sigma_{jr}})^2)(\frac{x_j-\bar{x}_{jr}}{\sigma_{jr}}) \tag{7}$$

where x_j and y_j are the fuzzy system input and output, respectively.

The parameters of the fuzzy systems have to be also adjusted. So, the sensitivity of the error with respect to the fuzzy systems parameters computed as

$$\begin{aligned}
\frac{\partial e_i}{\partial \bar{y}_{jr}}&=\frac{\partial e_i}{\partial z_i}\frac{\partial z_i}{\partial net_i}\frac{\partial net_i}{\partial y_j}\frac{\partial y_j}{\partial \bar{y}_{jr}}=\frac{\partial y_j}{\partial \bar{y}_{jr}}\sum_i \delta_i w_{ij}\\
\frac{\partial e_i}{\partial \bar{x}}&=\frac{\partial y_j}{\partial \bar{x}_{jr}}\sum_i \delta_i w_{ij}\\
\frac{\partial e_i}{\partial \sigma_{jr}}&=\frac{\partial y_j}{\partial \sigma_{jr}}\sum_i \delta_i w_{ij}
\end{aligned} \tag{8}$$

The derivatives ($\frac{\partial y_j}{\partial \bar{y}_{jr}}$, $\frac{\partial y_j}{\partial \bar{x}_{jr}}$ and, $\frac{\partial y_j}{\partial \sigma_{jr}}$) can be found in [8].

After computing the Jacobian matrix J the FNN parameters (p) which consist of the weigths(w_{jk}, v_{ij},) and fuzzy systems parameters ($\bar{y}_{jr}$, $\bar{x}_{jr}$, σ_{jr}) are updated with the LM method as

$$\begin{aligned}
(J_n^T J_n+\eta_n I)\Delta p_n&=-J_n^T e_n\\
p_{n+1}&=p_n+\Delta p_n
\end{aligned} \tag{9}$$

where $\eta_k \geq 0$ is a scalar, n is the epoch number, and I is the unit matrix. The trust region approach of Fletcher is used to determine η at every epoch [7].

4 Simulation Results

In this section, the simulation results of the identification of a nonlinear plant using the FNN are presented. The plant to be identified is a bioreactor. The dynamical equations of the bioreactor are given by [9]

$$\begin{aligned}
\frac{dc_1}{dt}&=-c_1 u+c_1(1-c_2)e^{c_2/\gamma}\\
\frac{dc_2}{dt}&=-c_2 u+c_1(1-c_2)e^{c_2/\gamma}(1+\beta)/(1+\beta-c_2)
\end{aligned} \tag{10}$$

where the state variables c_1 and c_2 are, respectively, dimensionless cell mass and substrate conversion, u is the flow rate through the bioreactor, β=0.02 is the cell growth rate constant, and γ=0.48 is the nutrient consumption constant. The simulation data sets are obtained integrating the dynamics equations of the bioreactor. They are integrated with a fourth-order Runge-Kutta algorithm using an integration step size of

Δ=0.01 time units. We define 50Δ as a macro time step. We generated the training and the test data sets by applying the different series of step signal input to the bioreactor model. The series-parallel identification model with three inputs which are the plant input and the past values of the plant states was set. The FNN has six neurons with fuzzy activation functions in he hidden layer and two linear neurons in the output layer. The fuzzy systems with three rules are used as the activation functions. The initial parameters of the fuzzy systems are chosen so that the membership functions uniformly cover the interval [-2,2] at the input and [-1,1] at the output. The Levenberg-Marquardt optimization method with the trust region approach of Fletcher is used to update network parameters.

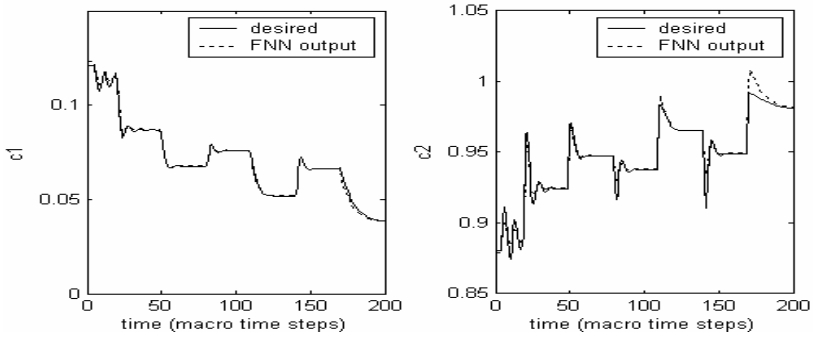

Fig. 1. The identification performance of the FNN

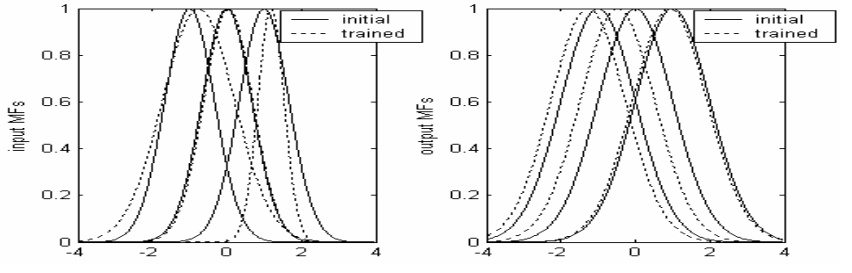

Fig. 2. The evolutions of the fuzzy rules with the training

The identification performance of the FNN is depicted in Fig. 1. The FNN outputs track the bioreactor states very well. The evolutions of the fuzzy rules with the training are shown in Fig. 1. The center and the spread of the input membership function and the center of the output membership function are adjusted with training. We can generate the IF-THEN rules from the trained fuzzy systems in the hidden layer. We define the Gaussian membership functions ($\mu_{A_j^r}(x_j;\overline{x}_{jr},\sigma_{jr})$ and $\mu_{B_j^r}(y_j;\overline{y}_{jr},1)$) with the center and spread parameters. R_j^r represents the r^{th} rule of the j^{th} neuron. As an example, the rules of the 1^{nd} neuron are generated as

$$
\begin{aligned}
&R_1^1: \quad \text{IF} \quad x_1 = A_1^1 \quad \text{THEN} \quad y_1 = B_1^1 \\
&\text{where} \quad \mu_{A_1^1}(x_1;-0.7622,0.9677), \quad \mu_{B_1^1}(y_1;-1.2727,1) \\
&R_1^2: \quad \text{IF} \quad x_1 = A_1^2 \quad \text{THEN} \quad y_1 = B_1^2 \\
&\text{where} \quad \mu_{A_1^2}(x_1;-0.0383,0.6575), \quad \mu_{B_1^2}(y_1;-0.4795,1) \\
&R_1^3: \quad \text{IF} \quad x_1 = A_1^3 \quad \text{THEN} \quad y_1 = B_1^3 \\
&\text{where} \quad \mu_{A_1^3}(x_1;1.1857,0.3532), \quad \mu_{B_1^3}(y_1;0.9226,1)
\end{aligned}
\tag{11}
$$

5 Conclusions

We have proposed to combine the fuzzy system paradigm with the MLP. The resulting FNN was designed by replacing the sigmoid type activation function of the MLP with the fuzzy system. We adapted the LM optimization method with the trust region approach of Fletcher for the fast update of the FNN parameters. A typical problem in ANNs was simulated in order to show the validity of the approach. A good tracking performance is obtained.

The IF-THEN rules from the trained fuzzy systems in the hidden layer have been generated. We can conclude that interpreting these rules together with the rule reduction techniques may lead to optimize the number of neurons in the hidden layer

References

1. Yamada, T., and Yabuta, T.: Neural Network Controller using Autotuning Method for Nonlinear Functions. IEEE Transactions on Neural Networks.(1992), 3, 595-601.
2. Chen, C.T. and Chang, W.D.: A Feedforward Neural Network with Function Shape Autotuning. Neural Networks, (1996) 9(4), 627-641.
3. Guarnieri, S., and Pizza. F.: Multilayer Feedforwardt Network with Adaptive Spline Activation Function. IEEE Transsactions on Neural Networks, (1999), 10(3), 672-683.
4. Trentin, E.: Networks with Trainable Amplitude of Activation Functions. Neural Networks, (2001), 14(4/5), 471-493.
5. Olivas, E.S., Guerrero, J.D.M., Valls, G.C., Lopez, A.J.S., Maravilla, J.C., and Chova, L.G.: A Low-Complexity Fuzzy Activation Function for Artificial Neural Networks. IEEE Transsactions on Neural Networks, (2003), 14(6), 1576-1579.
6. Oysal, Y., Becerikli, Y., and Konar, A.F.: Generalized modeling Principles of A Nonlinear System with a Dynamic Fuzzy Network. Computers & Chemical Engineering. 27 (2003), 1657-1664.
7. Scales, L.E.: Introduction to Non-Linear Optimization. Springer-Verlag New York Inc., USA, (1985) 115-118.
8. Wang, L.X.: A Course in Fuzzy Systems and Control. Prentice-Hall, Inc., (1997)
9. Ungar, L.H.: A Bioreactor Benchmark for Adaptive Network-based Process Control. In Neural Networks for Control, eds. Miller III, W.T., Sutton, R.S., and Werbos, P.J., MIT Press, London, (1990), 387-402.