The Third AFSS(1998). 387－390

Tuning Fuzzy Rules Based on Additive-Type Fuzzy System Models

Yan SHI[1] Masaharu MIZUMOTO[2]

[1]School of Engineering, Kyushu Tokai University
9-1-1, Toroku, Kumamoto 862-8652, Japan
Tel: +81-96-386-2666 Fax: +81-96-381-7956 E-mail: shi@ktmail.ktokai-u.ac.jp
[2]Division of Information and Computer Sciences, Osaka Electro-Communication University
18-8, Hatsu-cho, Neyagawa, Osaka 572-8530, Japan
Tel: +81-720-20-4569 Fax: +81-720-24-0014 E-mail: mizumoto@mzlab.osakac.ac.jp

Abstract

In this paper, we suggested a neuro-fuzzy learning algorithm for tuning fuzzy rules, in which a fuzzy system model is of additive-type. Using the method, it is possible to reduce the computation size, since performing the fuzzy inference and tuning the fuzzy rules for each fuzzy subsystem model are independent. Moreover, the efficiency of suggested method is shown by means of a numerical example.

Keywords: Additive-type fuzzy system, Neuro-fuzzy learning algorithm, Tuning fuzzy rules

1. Introduction

In recent fuzzy applications, in order to construct an optimal fuzzy system model to identify a practical problem, it is getting more than important to generate or tune fuzzy rules by means some of learning techniques, such as neuro-fuzzy methods proposed by Ichihashi [2], Shi *et al.* [4,5], Wang and Mendel [6]. However, when we deal with a large scale fuzzy system model, learning fuzzy rules is sometimes hard by using the conventional learning methods directly, because of the complexity of the fuzzy system model and the great number of the tuning parameters. In this paper, we suggest a neuro-fuzzy learning algorithm, in which the fuzzy system model is of additive-type. Using the method, it is possible that the computation size will become small, since performing the fuzzy inference and tuning the fuzzy rules for each fuzzy subsystem model are independent. Also, we show the efficiency of suggested method by a numerical example.

2. Additive-type fuzzy system models

First we briefly introduce so-called additive-type fuzzy system models.

Definition 1: Let f be a mapping from the input universe $X (= X_1 \times X_2 \times ... \times X_n)$ to the output universe Y, if there exist two sub-mappings f_1: $X_1 \times ... \times X_k \mapsto Y_1$ and f_2: $X_{k+1} \times ... \times X_n \mapsto Y_2$, such that f can be expressed a sum of f_1 and f_2, then we call $y^* = f(x_1, x_2, ... x_n)$ ($x_i \in X_i$, i=1,2,...,n; $y^* \in Y$) an additive-type system, which can be written as:

$$y^* = f(x_1,x_2,...,x_n) = f_1(x_1,x_2,...,x_k) + f_2(x_{k+1},x_{k+2},...,x_n) \quad (1)$$

Let x_1, x_2, ...,x_n be input variables on X and y be an output variable on Y, then fuzzy inference rules corresponding to the desired system $y^* = f(x_1,x_2,...,x_n)$ are arranged as follows:

$$\text{Rule } j1,...,(j,n-1),jn: A_1^{j1}, ..., A_{n-1}^{j,n-1}, A_n^{jn} \Rightarrow y_{(j1,...,(j,n-1),jn)} \quad (2)$$

where A_i^{ji} ($ji \in \{1,2,...,ki\}$, i=1,2,...,n) is a fuzzy set on the i-th input space X_i. $y_{(j1,...,(j,n-1),jn)}$ is a real number on the output universe Y, and ki (i=1,2,...,n) is the number of fuzzy partitions for the input variable x_i.

When an observation $(x_1,x_2,...,x_n)$ is given, a fuzzy inference consequence y can be obtained by using simplified fuzzy reasoning method as:

$$y = \frac{\sum_{j1=1}^{k1} ... \sum_{ji=1}^{ki} ... \sum_{jn=1}^{kn} A_1^{j1}(x_1)...A_i^{ji}(x_i)...A_n^{jn}(x_n) y_{(j1,...,(jn-1),jn)}}{\sum_{j1=1}^{k1} ... \sum_{ji=1}^{ki} ... \sum_{jn=1}^{kn} A_1^{j1}(x_1)...A_i^{ji}(x_i)...A_n^{jn}(x_n)} \quad (3)$$

Definition 2: If a given system $y^* = f_1(x_1,x_2,...,x_k) + f_2(x_{k+1},x_{k+2},...,x_n)$ is of additive-type, then we call the corresponded fuzzy system made by Eq.(2) and Eq.(3) an additive-type fuzzy system.

For such a additive-type fuzzy system model , we have the following result, which has been proved by Shi *et al.* [3], Yam [7].

Theorem 1: If a fuzzy system model is of additive-type, then the fuzzy inference consequence y in Eq.(3) can be obtained as follows:

$$y = \frac{\sum_{j1=1}^{k1} ... \sum_{jk=1}^{kk} A_1^{j1}(x_1)...A_k^{jk}(x_k) y^1_{(j1,...,jk)}}{\sum_{j1=1}^{k1} ... \sum_{jk=1}^{kk} A_1^{j1}(x_1)...A_k^{jk}(x_k)}$$

$$+\frac{\sum_{j,k+1=1}^{k,k+1}\cdots\sum_{jn=1}^{kn}A_{k+1}{}^{j,k+1}(x_{k+1})\ldots A_n{}^{jn}(x_n)y^2{}_{((j,k+1),\ldots,jn)}}{\sum_{j,k+1=1}^{k,k+1}\cdots\sum_{jn=1}^{kn}A_{k+1}{}^{j,k+1}(x_{k+1})\ldots A_n{}^{jn}(x_n)}$$

$$= I_1 + I_2 \qquad (4)$$

where I_i (i=1,2) is called a fuzzy subsystem model, $y^1_{(j1,\ldots,jk)}$ is a real number of the consequent parts corresponding to the fuzzy subsystem model I_1, and $y^2_{((j,k+1),\ldots,jn)}$ is a real number of the consequent parts corresponding to the fuzzy subsystem model I_2, subject to the condition $y_{(j1,\ldots,(j,n-1),jn)} = y^1_{(j1,\ldots,jk)} + y^2_{((j,k+1),\ldots,jn)}$.

By using the **Theorem 1**, one can see that the additive-type fuzzy system model becomes simple, and it is possible to perform a fuzzy inference for each dissolved fuzzy subsystem model independently, so that the computation size will become small, and that, it is also convenient for the user to design such fuzzy system model in the practical applications.

3. Tuning fuzzy rules based on an additive-type fuzzy system model

In this section, we shall derive a neuro-fuzzy learning algorithm for tuning fuzzy rules under an additive-type fuzzy system model with Gaussian-type membership functions, based on gradient descent method [1].

For the sake of technical simplification, we assume that the identified system has four input variables and one output variable, such as $y^* = f_1(x_1,x_2) + f_2(x_3,x_4)$, then by the **Theorem 1**, a fuzzy inference consequence y can be obtained as follows:

$$y = \frac{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1{}^i(x_1)A_2{}^j(x_2)y^1{}_{(i-1)k2+j}}{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1{}^i(x_1)A_2{}^j(x_2)}$$

$$+\frac{\sum_{m=1}^{k3}\sum_{n=1}^{k4}A_3{}^m(x_3)A_4{}^n(x_4)y^2{}_{(m-1)k4+n}}{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1{}^i(x_1)A_2{}^j(x_2)}$$

$$= I_1 + I_2 \qquad (5)$$

In this situation, the neural networks made by the additive-type fuzzy system is simple, for example, as shown in Fig. 1 when k1 = k2 = k3 = k4 = 2, where h_i^j (i=1,...,4; j=1,2) denotes an agreement of the antecedent parts corresponding to the fuzzy subsystem mode I_j (j=1,2), and y means a fuzzy inference consequence. On the other hand, in the case of an usual system $y^* = f(x_1,x_2,x_3,x_4)$ with the same membership functions as well as Fig. 1, the conventional neural networks made by a fuzzy system model can be expressed as shown in Fig. 2, where, h_k (k=1,2,...,16) stands for an agreement of the antecedent parts.

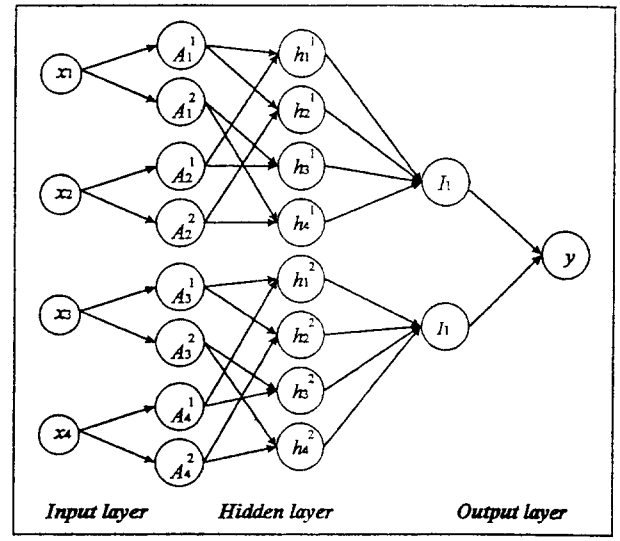

Fig. 1 Neural networks by an additive-type fuzzy system

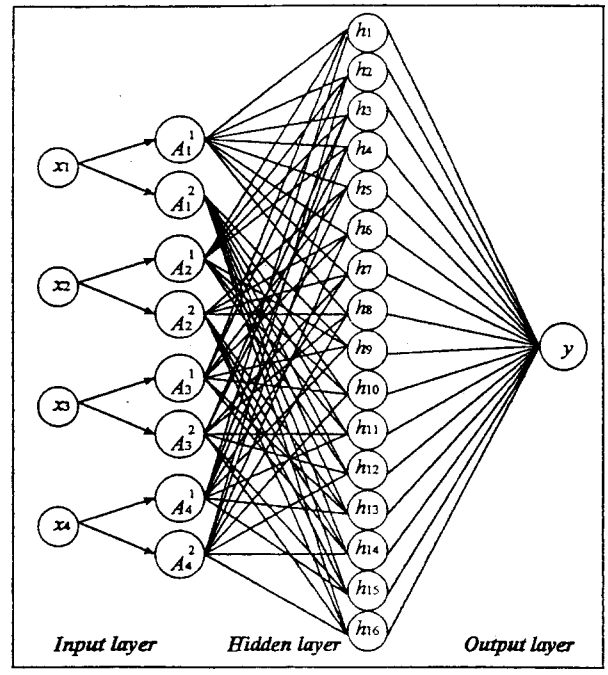

Fig. 2 Neural networks by a normal fuzzy system

To compare Fig. 1 with Fig. 2, one can see that the constructions of two kinds of neuro-fuzzy approaches are very different, so we can say that the suggested fuzzy inference approach is useful tool for designing a simple neuro-fuzzy system model.

Moreover, Gaussian-type membership functions on the antecedent parts are defined as:

$$A_i^{ji}(x_i) = exp(-(x_i - a_i^{ji})^2/b_i^{ji}) \qquad (6)$$

where a_1^{ji} (i=1,2,3,4; ji$\in${1,2,...,ki}) is the center of A_1^{ji}, b_1^{ji} means the width of A_1^{ji}.

When training input-output data (x_1,x_2,x_3,x_4; y^*) are given for the fuzzy system model, we adopt the following objective function E to evaluate the error between y^* and y:

$$E = (y^* - y)^2 / 2 \tag{7}$$

where y^* is a desired output value, and y is a fuzzy inference result.

In order to minimize the objective function E, based on gradient descent method [1] and the method by Shi *et al* [4,5], a learning algorithm for updating the parameters a_1^i, b_1^i, a_2^j, b_2^j and $y^1_{(i-1)k2+j}$ (i=1,2,...,k1; j=1,2,...,k2) is derived as follows:

$$a_1^i(t+1) = a_1^i(t) - \alpha\,\partial E/\partial a_1^i(t)$$
$$= a_1^i(t) - \alpha(\partial E/\partial y)(\partial y/A_1^i)(\partial A_1^i/\partial a_1^i(t))$$
$$= a_1^i(t) + 2\alpha(y^*-y)\frac{\sum_{j=1}^{k2}(y_{(i-1)k2+j}-I_1)A_2^j(x_2)}{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1^i(x_1)A_2^j(x_2)}\frac{A_1^i(x_1)(x_1-a_1^i(t))}{b_1^i(t)} \tag{8}$$

$$b_1^i(t+1) = b_1^i(t) - \beta\,\partial E/\partial b_1^i(t)$$
$$= b_1^i(t) - \beta(\partial E/\partial y)(\partial y/A_1^i)(\partial A_1^i/\partial b_1^i(t))$$
$$= b_1^i(t) + 2\beta(y^*-y)\frac{\sum_{j=1}^{k2}(y_{(i-1)k2+j}-I_1)A_2^j(x_2)}{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1^i(x_1)A_2^j(x_2)}\frac{A_1^i(x_1)(x_1-a_1^i(t))^2}{b_1^i(t)^2} \tag{9}$$

$$a_2^j(t+1) = a_2^j(t) - \alpha\,\partial E/\partial a_2^j(t)$$
$$= a_2^j(t) - \alpha(\partial E/\partial y)(\partial y/A_2^j)(\partial A_2^j/\partial a_2^j(t))$$
$$= a_2^j(t) + 2\alpha(y^*-y)\frac{\sum_{i=1}^{k1}(y_{(i-1)k2+j}-I_1)A_1^i(x_1)}{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1^i(x_1)A_2^j(x_2)}\frac{A_2^j(x_2)(x_2-a_2^j(t))}{b_2^j(t)} \tag{10}$$

$$b_2^j(t+1) = b_2^j(t) - \beta\,\partial E/\partial b_2^j(t)$$
$$= b_2^j(t) - \beta(\partial E/\partial y)(\partial y/A_2^j)(\partial A_2^j/\partial b_2^j(t))$$
$$= b_2^j(t) + 2\beta(y^*-y)\frac{\sum_{i=1}^{k1}(y_{(i-1)k2+j}-I_1)A_1^i(x_1)}{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1^i(x_1)A_2^j(x_2)}\frac{A_2^j(x_2)(x_2-a_2^j(t))^2}{b_2^j(t)^2} \tag{11}$$

$$y^1_{(i-1)k2+j}(t+1) = y^1_{(i-1)k2+j}(t) - \gamma\,\partial E/\partial y^1_{(i-1)k2+j}(t)$$
$$= y^1_{(i-1)k2+j}(t) - \gamma(\partial E/\partial y)(\partial y/\partial y^1_{(i-1)k2+j}(t))$$
$$= y^1_{(i-1)k2+j}(t) + \frac{\gamma(y^*-y)A_1^i(x_1)A_2^j(x_2)}{\sum_{i=1}^{k1}\sum_{j=1}^{k2}A_1^i(x_1)A_2^j(x_2)} \tag{12}$$

where α, β and γ are the learning rates, and t means the learning iteration.

Similarly, we can derive a learning algorithm for updating the parameters a_3^m, b_3^m, a_4^n, b_4^n and $y^2_{(m-1)k4+n}$ (m=1,2,...,k3; n=1,2,...,k4) in the same way.

4. Numerical example

In the sequel, we apply the suggested learning approach to the following nonlinear additive-type system with four input variables and one output variable for identifying and evaluating the problem, and show the efficiency of the suggested method.

Example:

$$y^* = (2x_1+4x_2^2+0.1)^2/37.21+\{[3exp(3x_3) + 2\,exp(-4x_4)]^{-1/2}-0.077\}/4.68 \tag{13}$$

where $x_i\in[-1,1]$ (i=1,...,4) is the input variable, and $y\in[0,1]$ is a normalized output variable.

To construct a corresponded additive-type fuzzy system model for the **Example**, first we assume that there exist five Gaussian-type membership functions on each input space, and the real numbers $y^1_{(i-1)5+j}$ and $y^2_{(m-1)5+n}$ (i,j,m,n=1,...,5) are set as shown in Eq.(14), subject to the condition $y_{(i,j,m,n)} = y^1_{(i-1)5+j} + y^2_{(m-1)5+n}$.

$$y^1_{(i-1)5+j} = 0, \quad y^2_{(m-1)5+n} = 0 \tag{14}$$

Then, 20 training data (x_1,x_2,x_3,x_4; y^*) are employed randomly for identifying the **Example** by using the learning algorithm Eqs.(8)-(12). In our case, the learning rates are taken as $\alpha = \beta = 0.05$, and $\gamma = 0.65$. The learning process is stopped when the inference error for D identifying data is less than the threshold δ. In this case,δ is taken as 0.005. Here, D is defined as follows:

$$D = \sum_{d=1}^{20}(y^*_d - y_d)^2/2 \tag{15}$$

where y^*_d (d=1,...,20) is a desired output value, and y_d is a fuzzy inference value.

Table 1 shows fuzzy inference results and desired output values for given 20 training data (x_1,x_2,x_3,x_4; y^*) and Table 2 shows fuzzy inference results and desired output values for given 20 checking data (x_1,x_2,x_3,x_4; y^*) to identify the **Example** by using the fuzzy rules generated by the suggested learning approach. In our case, 20 checking data are employed randomly.

In Table 2, the mean square error is 0.0063, the maximum absolute error is 0.1618, which shows finer approximate results by using the suggested learning approach for the given checking data. Therefore, we can say that the suggested approach is an efficient learning algorithm for tuning fuzzy rules when the fuzzy system model is of additive-type.

5. Conclusions

We have suggested an efficient neuro-fuzzy learning approach for tuning fuzzy rules when a fuzzy system model is of additive-type. By the method, it is possible to reduce the computation size in the neural processing system of fuzzy applications.

References

1 J.E. Dayhoff, Neural Network Architectures: An Introduction, Van Nostrand Reinhold, New York, 1990.

2 H. Ichihashi, "Iterative fuzzy modeling and a hierarchical network", Proc. of 4th IFSA Congress, 49-52, Brussels, 1991.

3 Y. Shi, M. Mizumoto, N. Yubazaki and M. Otani, "Fuzzy inference and fuzzy rule tuning for a dissolvable-type fuzzy system model", Proc. of the 12th Fuzzy System Symposium, 175-178, Tokyo, 1996.

4 Y. Shi, M. Mizumoto, N. Yubazaki and M. Otani, "A method of fuzzy rules generation based on neuro- fuzzy learning algorithm", Journal of Japan Society for Fuzzy Theory and Systems, 8(4), 695-705, 1996.

5 Y. Shi, M. Mizumoto, N. Yubazaki and M. Otani, "A learning algorithm for tuning fuzzy rules based the gradient descent method", Proc. the 5th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'96), 55-61, New Orleans, 1996.

6 L.X. Wang and J.M. Mendel, "Back-propagation fuzzy system as nonlinear dynamic system identifiers", Proc. of the IEEE International Conference on Fuzzy Systems, 1409-1416, San Diego, 1992.

7 Y. Yam, "Subsystem inference representation for fuzzy systems based upon product-sum-gravity rule", IEEE Trans. on Fuzzy Systems, 5 (1), 90-107, 1997.

Table 1 Training data for Identifying the **Example**

No.	x_1	x_2	x_3	x_4	y^*	y
1	-0.24	-0.62	0.50	1.00	0.1470	0.1396
2	-0.18	0.20	-0.12	-0.72	0.2465	0.2313
3	-0.14	0.20	0.64	0.86	0.2772	0.2904
4	-0.92	0.42	-0.80	-0.92	0.2368	0.2371
5	0.14	-0.32	-0.38	0.98	0.5629	0.5643
6	-0.48	0.72	-0.14	-0.52	0.0645	0.0798
7	0.98	-0.50	-0.36	-0.96	0.2660	0.2690
8	-0.78	0.84	0.44	-0.92	0.0767	0.0731
9	-0.42	-0.78	0.56	-0.10	0.0568	0.0609
10	0.48	0.20	-0.18	-0.18	0.5556	0.5561
11	0.34	-0.86	-0.80	-0.24	0.3953	0.3956
12	0.24	0.46	-0.50	0.12	0.5128	0.5108
13	-0.66	-0.58	0.72	0.22	0.1080	0.1016
14	0.90	-0.72	0.06	-0.70	0.2673	0.2622
15	-0.62	-0.82	0.22	-0.88	0.0332	0.0460
16	-0.84	-0.54	0.00	-0.78	0.1736	0.1723
17	0.86	-0.14	0.74	0.92	0.4205	0.4179
18	-0.56	-0.76	0.78	-0.90	0.0300	0.0290
19	-0.74	0.46	-0.66	0.00	0.2600	0.2624
20	-0.86	-0.18	0.08	-0.32	0.2974	0.2961

Table 2 Checking data for Identifying the **Example**

No.	x_1	x_2	x_3	x_4	y^*	y
1	-0.96	0.30	-0.26	0.68	0.4357	0.2739
2	0.36	0.22	0.14	-0.26	0.5154	0.4911
3	-0.02	-0.14	0.56	0.16	0.3501	0.2974
4	-0.98	0.16	0.08	-0.24	0.3672	0.3463
5	-0.44	-0.74	0.56	-0.90	0.0354	0.0330
6	0.40	0.08	0.18	0.12	0.5573	0.5549
7	-0.44	0.30	-0.50	-0.94	0.1418	0.0918
8	-0.54	0.06	0.92	-0.52	0.1769	0.0885
9	0.30	-0.88	-0.84	-0.20	0.3895	0.3798
10	-0.24	0.82	0.00	0.74	0.1704	0.1039
11	0.66	0.62	0.50	-0.34	0.3953	0.2568
12	0.42	0.64	-0.06	0.92	0.4931	0.3739
13	0.20	-0.18	-0.80	-0.92	0.4258	0.3963
14	-0.60	-0.64	-0.58	0.50	0.2789	0.1216
15	-0.44	0.74	-0.58	-0.46	0.0718	0.0546
16	-0.88	0.00	-0.64	-0.28	0.3389	0.2932
17	-0.78	0.26	-0.32	-0.58	0.2304	0.1433
18	0.16	-0.88	0.86	0.42	0.2702	0.2555
19	0.26	-0.46	-0.86	-0.48	0.4229	0.3686
20	-0.72	0.90	-0.36	0.86	0.2312	0.3232