2013 Second International Conference on Robot, Vision and Signal Processing

Quaternion Neuro-Fuzzy Learning Algorithm for Fuzzy Rule Generation

Ryusuke Hata
Graduate School of Engineering
University of Fukui
Fukui, Japan
hata.r.1324@gmail.com

Md. Monirul Islam
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

Kazuyuki Murase
Graduate School of Engineering
University of Fukui
Fukui, Japan
murase@u-fukui.ac.jp

***Abstract*—In order to generate or tune fuzzy rules, Neuro-Fuzzy learning algorithms with Gaussian type membership functions based on gradient-descent method are well known. In this paper, we propose a new learning approach, the Quaternion Neuro-Fuzzy learning algorithm. This method is an extension of the conventional method to four-dimensional space by using a quaternion neural network that maps quaternion to real values. Input, antecedent membership functions and consequent singletons are quaternion, and output is real. Four-dimensional input can be better represented by quaternion than by real values. We compared it with the conventional method by several function identification problems, and revealed that the proposed method outperformed the counterpart: The number of rules was reduced to 5 from 625, the number of epochs by one fortieth, and error by one tenth in the best cases.**

Keywords-neuro-fuzzy; quaternion neural networks; fuzzy; neural networks

I. INTRODUCTION

In the field of fuzzy control, the practical applications of fuzzy inference have increased, and generations of fuzzy rules have become important. These include tuning of membership functions and rules. However, when a fuzzy system model is designed, it is sometimes too hard or impossible for human beings to give the desired fuzzy rules, due to the ambiguity, uncertainty or complexity of the identifying system. Many methods have been constructed by combining fuzzy systems and neural networks to generate or tune fuzzy rules of fuzzy system models [1]. These methods, called Neuro-Fuzzy learning algorithms (NFs), recently have been successfully applied to, e.g. control system and system identification. Further, a variety of system structures and learning algorithms are available for NFs.

In this paper, we use a method of tuning fuzzy rules and its parameters by back propagation learning algorithm of neural networks [1]. Such NFs, whose antecedent membership function is fixed for each fuzzy inference rule under the simplified fuzzy inference method, can generate fuzzy rules by automatic tuning of its parameters and the consequent singleton values based on a gradient-descent method. However, if we use multi-input for this method, the number of parameter of antecedent membership functions increase rapidly with increasing the number of fuzzy inference rules. For this reason, it takes a long period of time for learning and the learning accuracy may deteriorate [2].

We focused on Quaternion Back Propagation (QBP) [3] of Quaternion Neural Networks (QNNs). QNN is an extension of Real-valued Neural Networks (RVNNs) and has better learning ability than RVNNs [3]. Further, QNN has been applied to time series prediction, rigid control and color night vision [4].

In this paper, we propose the Quaternion Neuro-Fuzzy learning algorithm (QNF). It extends the antecedent membership function and the consequent singleton of the conventional method to four-dimensional space and generates real-valued output for quaternion inputs. Further, we compared it with the conventional method by several function identification problems, and show the superiority.

II. NF AND QNF

III. *Conventional NF*

In the conventional NF, if the inputs are x_i (i = 1, 2, ⋯, n) and the output is Y, then fuzzy inference rules of the simplified fuzzy inference are shown below:

Rule 1: If x_1 *is* M_{11} *and* x_2 *is* M_{12} *...* x_n *is* M_{1n}

Then Y *is* W_1

Rule 2: If x_1 *is* M_{21} *and* x_2 *is* M_{22} *...* x_n *is* M_{2n}

Then Y *is* W_2

...

Rule m: If x_1 *is* M_{m1} *and* x_2 *is* M_{m2} *...* x_n *is* M_{mn}

Then Y *is* W_m (1)

where W_j (j=1, 2, ⋯, m) are real value of the consequent singleton.

The antecedent membership functions M_{ji} (j = 1, 2, ⋯, m; i = 1, 2, ⋯, n) are given by Gaussian function as,

978-1-4799-3184-2/13 $31.00 © 2013 IEEE
DOI 10.1109/RVSP.2013.22

$$M_{ji}(x_i) = exp\left(-\left(x_i - a_{ji}\right)^2 / b_{ji}\right) \quad (2)$$

The inference result Y is as follows. First, the grade of the antecedent is given by

$$R_j = \prod_{i=1}^{n} M_{ji}(x_i) \qquad (j = 1,\ 2,\ \ldots,\ m) \quad (3)$$

Then, the inference result Y is calculated by the following gravity method.

$$Y = \frac{\sum_{j=1}^{m} R_j W_j}{\sum_{j=1}^{m} R_j} \quad (4)$$

The error function to be minimized during the training is given by

$$E = \frac{1}{2}(Y - T)^2 = \frac{1}{2}e^2 \quad (5)$$

where T is the desired output. During the training, each parameter W_j, a_{ji}, b_{ji} is updated by,

$$\Delta W_j = -\alpha \frac{\partial E}{\partial W_j} = -\alpha \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial W_j} = -\alpha e \frac{R_j}{\sum_{k=1}^{m} R_k} \quad (6)$$

$$\Delta a_{ji} = -\beta \frac{\partial E}{\partial a_{ji}} = -\beta \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial R_j}\frac{\partial R_j}{\partial M_{ji}}\frac{\partial M_{ji}}{\partial a_{ji}}$$

$$= -2\beta e R_j \frac{W_j - Y}{\sum_{k=1}^{m} R_k}\frac{x_i - a_{ji}}{b_{ji}} \quad (7)$$

$$\Delta b_{ji} = -\gamma \frac{\partial E}{\partial b_{ji}} = -\gamma \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial R_j}\frac{\partial R_j}{\partial M_{ji}}\frac{\partial M_{ji}}{\partial b_{ji}}$$

$$= -\gamma e R_j \frac{W_j - Y}{\sum_{k=1}^{m} R_k}\frac{\left(x_i - a_{ji}\right)^2}{b_{ji}^2} \quad (8)$$

where $\alpha,\ \beta\ and\ \gamma$ are the learning rate.

We can perform the learning process by giving the initial value to each parameter and by using Eq. (6) – (8).

IV. The QNF

In the learning algorithm we propose that each parameter is extended to a quaternion, and is given by the following flow.

Fuzzy inference rules are shown below:

Rule 1: If X_1 is M_{11} and X_2 is M_{12} ... X_n is M_{1n}

Then Y is W_1

Rule 2: If X_1 is M_{21} and X_2 is M_{22} ... X_n is M_{2n}

Then Y is W_2

...

Rule m: If X_1 is M_{m1} and X_2 is M_{m2} ... X_n is M_{mn}

Then Y is W_m (9)

where, the variables except Y are quaternion, Y is the real-valued output. $X_i = x_i^R + ix_i^I + jx_i^J + kx_i^K$ is the quaternion input. $M_{ji} = M_{ji}^R + iM_{ji}^I + jM_{ji}^J + kM_{ji}^K$ is the quaternion membership function. $W_j = W_j^R + iW_j^I + jW_j^J + kW_j^K$ is the quaternion singleton. Here, marks *R*, *I*, *J* and *K* denote one real and three imaginary parts of quaternion (same as above). Further, *i*, *j* and *k* in front of the variables represent imaginary numbers, and $i = j = k = \sqrt{-1}$. The antecedent membership functions are given by

$$M_{ji}^A(x_i^A) = exp\left\{-\left(x_i^A - a_{ji}^A\right)^2 / b_{ji}^A\right\} \quad (10)$$

where, marks *R*, *I*, *J* and *K* are given to mark *A* (same as above). This means that Gaussian function is given to the real and imaginary parts of the antecedent membership functions. The inference result Y is calculated as follows. First, the grade of the real and imaginary parts of the antecedent is given by

$$R_j^A = \prod_{i=1}^{n} M_{ji}^A(x_i^A) \quad (11)$$

Second, the quaternion inference result $Z = Z^R + iZ^I + jZ^J + kZ^K$ is calculated by the gravity method.

$$Z = \frac{\sum_{j=1}^{m}\left\{\left(R_j^R + iR_j^I + jR_j^J + kR_j^K\right)\left(W_j^R + iW_j^I + jW_j^J + kW_j^K\right)\right\}}{\sum_{j=1}^{m}\left(R_j^R + iR_j^I + jR_j^J + kR_j^K\right)} \quad (12)$$

Finally, the real-valued inference result Y is calculated as follows:

$$Y = f_{Q\to R}(Z) \quad (13)$$

$$f_{Q\to R}(Z) = \left(Z^R - Z^I\right)^2 + \left(Z^J - Z^K\right)^2 \quad (14)$$

where Eq. (14) is the activation function that we have newly developed. It is similar to the one used for complex-valued neural networks [5]. By this activation function, we are able to get the real-valued inference result *Y*.

The error function is the same as Eq. (5). During the training, each parameter is updated by,

$$\Delta W_j = -\alpha \frac{\partial E}{\partial w_j^R} - i\alpha \frac{\partial E}{\partial w_j^I} - j\alpha \frac{\partial E}{\partial w_j^J} - k\alpha \frac{\partial E}{\partial w_j^K} \quad (15)$$

$$\Delta a_{ji} = -\beta \frac{\partial E}{\partial a_{ji}^R} - i\beta \frac{\partial E}{\partial a_{ji}^I} - j\beta \frac{\partial E}{\partial a_{ji}^J} - k\beta \frac{\partial E}{\partial a_{ji}^K} \quad (16)$$

$$\Delta b_{ji} = -\gamma \frac{\partial E}{\partial b_{ji}^R} - i\gamma \frac{\partial E}{\partial b_{ji}^I} - j\gamma \frac{\partial E}{\partial b_{ji}^J} - k\gamma \frac{\partial E}{\partial b_{ji}^K} \quad (17)$$

where $\alpha,\ \beta\ and\ \gamma$ are the learning rate. Since Eq. (15) – (17) are not available directly, we need to expand each equation as follows (here, we present only the real part of each parameter).

$$\frac{\partial E}{\partial w_j^R} = \frac{\partial E}{\partial Y}\left(\frac{\partial Y}{\partial Z^R}\frac{\partial Z^R}{\partial w_j^R} + \frac{\partial Y}{\partial Z^I}\frac{\partial Z^I}{\partial w_j^R} + \frac{\partial Y}{\partial Z^J}\frac{\partial Z^J}{\partial w_j^R} + \frac{\partial Y}{\partial Z^K}\frac{\partial Z^K}{\partial w_j^R}\right) \quad (18)$$

$$\frac{\partial E}{\partial a_{ji}^R} = \frac{\partial E}{\partial Y}\frac{\partial R_j^R}{\partial M_{ji}^R}\frac{\partial M_{ji}^R}{\partial a_{ji}^R}\left(\frac{\partial Y}{\partial Z^R}\frac{\partial Z^R}{\partial R_j^R} + \frac{\partial Y}{\partial Z^I}\frac{\partial Z^I}{\partial R_j^R} + \frac{\partial Y}{\partial Z^J}\frac{\partial Z^J}{\partial R_j^R} + \frac{\partial Y}{\partial Z^K}\frac{\partial Z^K}{\partial R_j^R}\right) \quad (19)$$

$$\frac{\partial E}{\partial b_{ji}^R}=\frac{\partial E}{\partial Y}\frac{\partial R_j^R}{\partial M_{ji}^R}\frac{\partial M_{ji}^R}{\partial b_{ji}^R}\left(\frac{\partial Y}{\partial Z^R}\frac{\partial Z^R}{\partial R_j^R}+\frac{\partial Y}{\partial Z^I}\frac{\partial Z^I}{\partial R_j^R}+\frac{\partial Y}{\partial Z^J}\frac{\partial Z^J}{\partial R_j^R}+\frac{\partial Y}{\partial Z^K}\frac{\partial Z^K}{\partial R_j^R}\right) \tag{20}$$

Then, each partial differential of Eq. (18) – (20) is determined as follows.

$$\frac{\partial E}{\partial Y}=e \tag{21}$$

$$\frac{\partial Y}{\partial Z^R}=2(\mathrm{Z^R}-\mathrm{Z^I}) \tag{22}$$

$$\frac{\partial Y}{\partial Z^I}=-2(\mathrm{Z^R}-\mathrm{Z^I}) \tag{23}$$

$$\frac{\partial Y}{\partial Z^J}=2(\mathrm{Z^J}-\mathrm{Z^K}) \tag{24}$$

$$\frac{\partial Y}{\partial Z^K}=-2(\mathrm{Z^J}-\mathrm{Z^K}) \tag{25}$$

$$\frac{\partial Z}{\partial W_j^R}=\frac{\left(R_j^R+iR_j^I+jR_j^J+kR_j^K\right)\left(\sum_{k=1}^m R_k^*\right)}{\left(\sum_{k=1}^m R_k^R\right)^2+\left(\sum_{k=1}^m R_k^I\right)^2+\left(\sum_{k=1}^m R_k^J\right)^2+\left(\sum_{k=1}^m R_k^K\right)^2} \tag{26}$$

$$\frac{\partial Z}{\partial R_j^R}=\frac{\left(W_j^R+iW_j^I+jW_j^J+kW_j^K\right)\left(\sum_{k=1}^m R_k^*\right)+\left(\sum_{k=1}^m R_k W_k\right)-2R_j^R Z}{\left(\sum_{k=1}^m R_k^R\right)^2+\left(\sum_{k=1}^m R_k^I\right)^2+\left(\sum_{k=1}^m R_k^J\right)^2+\left(\sum_{k=1}^m R_k^K\right)^2} \tag{27}$$

$$\frac{\partial R_j^R}{\partial M_{ji}^R}=\frac{R_j^R}{M_{ji}^R} \tag{28}$$

$$\frac{\partial M_{ji}^R}{\partial a_{ji}^R}=2M_{ji}^R\left(x_i^R-a_{ji}^R\right)/b_{ji}^R \tag{29}$$

$$\frac{\partial M_{ji}^R}{\partial b_{ji}^R}=-M_{ji}^R\left(x_i^R-a_{ji}^R\right)^2/\left(b_{ji}^R\right)^2 \tag{30}$$

Where, R_k^* denotes the quaternion conjugate of R_k.

As same as the conventional method, we can perform the learning process by giving the initial value to each parameter and using Eq. (15) – (17).

V. Simulation Results

In the previous section, we proposed the QNF to get fuzzy rules, and presented its learning algorithm under Gaussian type membership functions. In this section, we compare it with the conventional method by several function identification problems, and show that the proposed method is a useful tool for learning a fuzzy system model.

Function Identifications

We take the following two nonlinear functions with four inputs and one output. Eq. (31) and (32) are quoted from a literature [6].

Function 1:

$$y=\frac{\left(2x_1+4x_2^2+0.1\right)^2}{74.42}+\frac{\left\{\left(3e^{3x_3}+2e^{-4x_4}\right)^{-0.5}-0.077\right\}}{4.68} \tag{31}$$

Function 2:

$$y=\frac{\left(2x_1+4x_2^2+0.1\right)^2}{37.21}\cdot\frac{4\sin(\pi x_3)+2\cos(\pi x_4)+6}{12} \tag{32}$$

Where, $x_1, x_2, x_3, x_4 \in [-1, 1]$ are the input variables, and $y \in [0, 1]$ is the output variable.

Then, using these two functions, we compare the new method with the conventional method about the epoch and the evaluation error when the number of rules is the same.

Tables 1 and 2 are the initial values of each parameter of each method. In two functions, for initialization, we divided each antecedent input space in five by Gaussian type membership functions. Accordingly, the conventional method, the number of fuzzy rule is twenty five. However, in the new method, we give x_1, x_2, x_3 and x_4 to one real and three imaginary parts of one input. Note that, in terms of the number of the fuzzy rules, the new method (5 rules) is smaller than the conventional method (625 rules).

Table 1. NF

M_{ji}	W_j
(-1, 0.12)	0.5
(-0.5, 0.12)	0.5
(0, 0.12)	0.5
(0.5, 0.12)	0.5
(1, 0.12)	0.5

Table 2. QNF

M_{ji}^A	$\left(W_j^R, W_j^I, W_j^J, W_j^K\right)$
(-1, 0.12)	(0.6, 0.5, 0.4, 0.3)
(-0.5, 0.12)	(0.6, 0.5, 0.4, 0.3)
(0, 0.12)	(0.6, 0.5, 0.4, 0.3)
(0.5, 0.12)	(0.6, 0.5, 0.4, 0.3)
(1, 0.12)	(0.6, 0.5, 0.4, 0.3)

In Eq. (33), E_{all} is the fuzzy inference error for the training set. Then, we applied both methods to Functions 1 and 2, and tuned the fuzzy rules until E_{all} becomes smaller than the threshold δ. The results are shown in Tables 3 and 4. Results shown are the average of 20 trials. In these Tables,

$$E_{all}=\frac{1}{2N}\sum_{d=1}^N(Y_d-T_d)^2 \tag{33}$$

where Y_d is the fuzzy inference, T_d is the desired output, and N is the number of training set.

In Tables 1 and 2, the training set is given by

Equivalent-81

$$x_1, x_2, x_3, x_4 \in \{-0.9, 0, 0.9\} \tag{34}$$

Equivalent-625

$$x_1, x_2, x_3, x_4 \in \{-0.9, -0.5, 0, 0.5, 0.9\} \tag{35}$$

Table 3. NF vs. QNF for Function 1

Function 1		NF: α=0.1, β=0.01, γ=0.01, QNF: α=0.1, β=0.01, γ=0.01								
Number of data	δ	No.	Epoch		Evaluation		Standard deviation		Max. absolute error	
			NF	QNF	NF	QNF	NF	QNF	NF	QNF
Random-81	0.003	①	5150	227	0.0229	**0.0044**	0.00007	0.00017	**0.4811**	0.6225
		②	5221	173	0.0259	**0.0036**	0.00008	0.00030	0.4889	**0.3235**
Equivalent-81	0.003	③	4709	287	0.0331	**0.0030**	0.00007	0.00006	0.4338	**0.3183**
Random-625	0.003	④	8709	197	0.0058	**0.0035**	0.00004	0.00009	**0.4402**	0.4945
		⑤	8963	215	0.0056	**0.0032**	0.00003	0.00010	**0.4388**	0.4682
Equivalent-625	0.003	⑥	9941	185	**0.0032**	0.0035	0.00003	0.00010	**0.3626**	0.4744

The evaluation error is given as follows. First, we perform learning each fuzzy rule by the conventional method and the new method. Second, we input 14641 evaluation data (x_1, x_2, x_3, x_4) (where these ranges of x_1, x_2, x_3 and x_4 are increments of 0.2 from -1 to 1) for Functions 1 and 2 to each learned fuzzy rule. Finally, we get the mean

squared error between its output and the desired output for Functions 1 and 2. This is the evaluation error.

Table 4. NF vs. QNF for Function 2

Function 2			NF: α=0.1, β=0.01, γ=0.01, QNF: α=0.1, β=0.01, γ=0.01							
Number of data	δ	No.	Epoch		Evaluation		Standard deviation		Max. absolute error	
			NF	QNF	NF	QNF	NF	QNF	NF	QNF
Random-81	0.003	①	5488	208	0.0332	**0.0040**	0.00009	0.00025	**0.5623**	0.5794
		②	5899	322	0.0350	**0.0028**	0.00009	0.00013	**0.4918**	0.5045
Equivalent-81	0.003	③	5014	160	0.0459	**0.0053**	0.00008	0.00025	**0.4830**	0.6485
Random-625	0.003	④	11275	347	0.0065	**0.0026**	0.00006	0.00011	**0.4169**	0.4776
		⑤	10791	305	0.0071	**0.0032**	0.00006	0.00009	**0.4144**	0.6141
Equivalent-625	0.003	⑥	11335	232	**0.0033**	0.0038	0.00002	0.00005	**0.4136**	0.6647

The evaluation error is given as follows. First, we perform learning each fuzzy rule by the conventional method and the new method. Second, we input 14641 evaluation data (x_1, x_2, x_3, x_4) (where these ranges of x_1, x_2, x_3 and x_4 are increments of 0.2 from -1 to 1) for Functions 1 and 2 to each learned fuzzy rule. Finally, we get the mean squared error between its output and the desired output for Functions 1 and 2. This is the evaluation error.

As an example, using the random data 2 in Table 4, we generated each fuzzy rule for the conventional method and the new method. Fig. 1 (a) and (b) are each result of the fuzzy inference for 14641 evaluation data. Fig. 2 (c) is the desired output of Function 2. Further, Fig. 2 (a) and (b) shows the absolute error between each result of the fuzzy inference and the desired output.

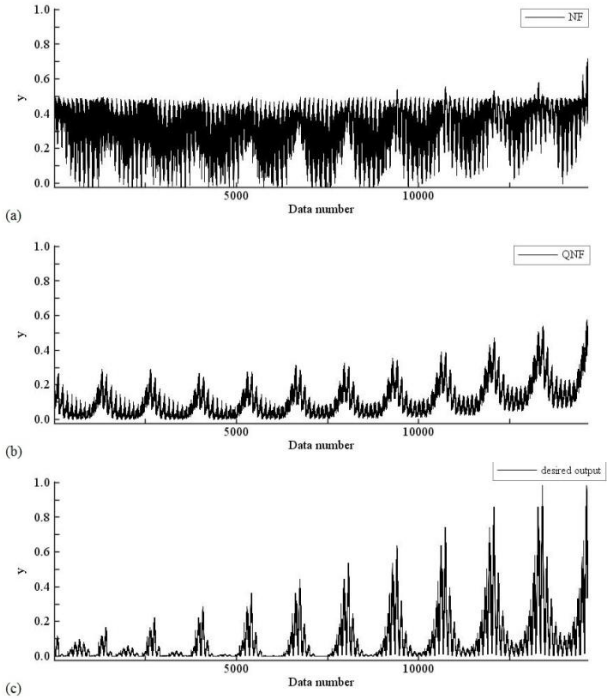

Fig. 1. Desired output and fuzzy inference for Function 2: (a) NF. (b) QNF. (c) Desired output for Function 2.

From Fig. 1 and 2, compared with the new method, the conventional method could not learn enough as a whole. Further, the new method could fit to such random training sets.

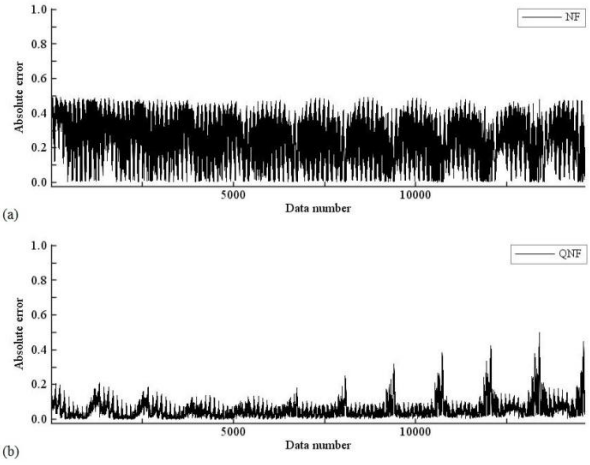

Fig. 2. Absolute error between desired output and fuzzy inference for Function 2: (a) NF. (b) QNF.

VI. DISCUSSION

By the analysis of the results shown in Tables 3 and 4, Fig. 1 and 2, we can describe as follows.

(1) The number of rules was drastically reduced from 625 to 5. And also, the number of epochs to converge was around one twentieth. We used a simple activation function, other types of activation functions need to be tested.

(2) In terms of the evaluation error, we found that the new method is much better than the conventional method for two functions. In particular, the evaluation error for random training sets showed good result for all functions. Thus, we can say that although the freedom of parameters is limited, the new method could fit for training sets well.

(3) In terms of the maximum absolute error, for both Functions 1 and 2, the conventional method showed slightly better results than the new method. We consider that this is because of the overfitting of the new method. However, from Fig. 2, we found that the new method, overall error is smaller than the conventional method.

From the above results of the simulation, we can conclude that the new method has equivalent to or better accuracy than the conventional method. Furthermore, the new method has a feature that while the parameters have less flexibility, it can fit for training sets well. Therefore, we can say that the new method is a useful tool for learning the fuzzy system model.

VII. SUMMARY

In this paper, we proposed the new method extending the conventional method to the four-dimensional space for tuning fuzzy rules. Then, we gave the general formulas for this algorithm under Gaussian type membership functions. Finally, in several function identification problems, we

showed that the new method outperforms the conventional approach for learning a fuzzy system model.

In the future, we like to show the effectiveness of the proposed method in the subject that can be represented by quaternion such as 3D-image and time series data. Furthermore, when we extend our algorithm for multivariate input, we consider proposing the hierarchical algorithm so as to suppress the increase of the number of the fuzzy rules, beside the use of hyper-complex numbers.

ACKNOWLEDGEMENTS

Supported by grants to KM from the Japanese Society for Promotion of Sciences and the University of Fukui.

REFERENCES

[1] Ichihashi, H.: Iterative fuzzy modeling and a hierarchical network. In: Proceedings of the Fourth IFSA Congress, Vol. Engineering, Brussels (1991) 49-52

[2] Shi, Y., Mizumoto, M., Yubazaki, N., & Otani, M.: A method of fuzzy rules generation based on neuro-fuzzy learning algorithm. J. Jpn Soc. Fuzzy Theory Systems, 8 (4) (1996) 695–705

[3] Matsui, N., Isokawa, T., Kusamichi, H., Peper, F., & Nishimura, H.: Quaternion neural network with geometrical operators. Journal of Intelligent and Fuzzy Systems, 15 (3) (2004) 149-164

[4] Kusamichi, H., Isokawa, T., Matsui, N., Ogawa, Y., & Maeda, K.: A new scheme for color night vision by quaternion neural network. In 2nd International Conference on Autonomous robots & agents (2004)

[5] Amin, M. F., Murase, K.: Single-layered complex-valued neural network for real-valued classification problems. Neurocomputing 72 (4) (2009) 945-955

[6] Yubazaki, N., Yi, J. & Hirota, K.: A Proposal of SIRMs (Single Input Rule Modules) Connected Fuzzy Inference Model for Plural Input Fuzzy Control. J. Jpn Soc. Fuzzy Theory Systems, 9 (5) (1997) 699-709