Quaternon Neuro-Fuzzy Learning Algorithm for Fuzzy Rule Generation

<table>
<thead>
<tr>
<th>メタデータ</th>
<th>言語: eng</th>
</tr>
</thead>
<tbody>
<tr>
<td>出版者:</td>
<td></td>
</tr>
<tr>
<td>公開日: 2014-07-30</td>
<td></td>
</tr>
<tr>
<td>キーワード (Ja):</td>
<td></td>
</tr>
<tr>
<td>キーワード (En):</td>
<td></td>
</tr>
<tr>
<td>作成者: Hata, Ryusuke, Islam, Monirul, Murase, Kazuyuki</td>
<td></td>
</tr>
<tr>
<td>メールアドレス:</td>
<td></td>
</tr>
<tr>
<td>所属:</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10098/8432">http://hdl.handle.net/10098/8432</a></td>
</tr>
</tbody>
</table>

(c) 2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
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 fourth, 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 \( x_i \) (\( i = 1, 2, \ldots, n \)) and the output is \( Y \), then fuzzy inference rules of the simplified fuzzy inference are shown below:

\[
\text{Rule 1: If } x_1 \text{ is } M_{11} \text{ and } x_2 \text{ is } M_{12} \text{ ... } x_n \text{ is } M_{1n} \\
\text{Then } Y \text{ is } W_1
\]

\[
\text{Rule 2: If } x_1 \text{ is } M_{21} \text{ and } x_2 \text{ is } M_{22} \text{ ... } x_n \text{ is } M_{2n} \\
\text{Then } Y \text{ is } W_2
\]

\[
\ldots
\]

\[
\text{Rule m: If } x_1 \text{ is } M_{m1} \text{ and } x_2 \text{ is } M_{m2} \text{ ... } x_n \text{ is } M_{mn} \\
\text{Then } Y \text{ is } W_m
\]

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

The antecedent membership functions \( M_{ij} \) (\( j = 1, 2, \ldots, m; \ i = 1, 2, \ldots, n \)) are given by Gaussian function as,
\[ M_{ji}(x_i) = \exp \left( -\frac{(x_i - a_{ji})^2}{b_{ji}} \right) \]  \hspace{1cm} (2)

The inference result \( Y \) is as follows. First, the grade of the antecedent is given by
\[ R_j = \prod_{i=1}^{n} M_{ji}(x_i) \quad (j = 1, 2, \ldots, m) \]  \hspace{1cm} (3)

Then, the inference result \( Y \) is calculated by the following gravity method.
\[ Y = \frac{\sum_{j=1}^{n} R_j W_j}{\sum_{j=1}^{n} R_j} \]  \hspace{1cm} (4)

The error function to be minimized during the training is given by
\[ E = \frac{1}{2} (Y - T)^2 = \frac{1}{2} \varepsilon^2 \]  \hspace{1cm} (5)

where \( T \) is the desired output. During the training, each parameter \( W_j, a_{ji}, b_{ji} \) is updated by,
\[ \Delta W_j = -\alpha \frac{\partial E}{\partial W_j} = -\alpha \frac{\partial E}{\partial Y} \frac{\partial Y}{\partial W_j} = -\alpha E \frac{R_j}{\sum_{k=1}^{n} R_k} \]  \hspace{1cm} (6)

\[ \Delta a_{ji} = -\beta \frac{\partial E}{\partial a_{ji}} = -\beta \frac{\partial E}{\partial Y} \frac{\partial Y}{\partial a_{ji}} = -2\beta E R_j \frac{w_{ji}Y}{\sum_{k=1}^{n} R_k} \frac{x_i - a_{ji}}{b_{ji}} \]  \hspace{1cm} (7)

\[ \Delta b_{ji} = -\gamma \frac{\partial E}{\partial b_{ji}} = -\gamma \frac{\partial E}{\partial Y} \frac{\partial Y}{\partial b_{ji}} = -\gamma E R_j \frac{w_{ji}Y}{\sum_{k=1}^{n} R_k} \frac{(x_i - a_{ji})^2}{b_{ji}^3} \]  \hspace{1cm} (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 \( W_m \)

where, the variables except \( Y \) are quaternion, \( Y \) is the real-valued output. \( x_i^{R} + ix_i^{I} + jx_i^{J} + kx_i^{K} \) 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
\[ M_{ji}^{R}(x_i^{R}) = \exp \left\{ -\frac{(x_i^{R} - a_{ji}^{R})^2}{b_{ji}^{R}} \right\} \]  \hspace{1cm} (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
\[ R_j^{A} = \prod_{i=1}^{n} M_{ji}^{A}(x_i^{A}) \]  \hspace{1cm} (11)

Second, the quaternion inference result \( Z = Z^{R} + iZ^{I} + jZ^{J} + kZ^{K} \) is calculated by the gravity method.
\[ Z = \frac{\sum_{j=1}^{m} \left( (R_j^{R} + iR_j^{I} + jR_j^{J} + kR_j^{K}) (W_j^{R} + iW_j^{I} + jW_j^{J} + kW_j^{K}) \right)}{\sum_{j=1}^{m} (R_j^{R} + iR_j^{I} + jR_j^{J} + kR_j^{K})} \]  \hspace{1cm} (12)

Finally, the real-valued inference result \( Y \) is calculated as follows:
\[ Y = f_{Q \rightarrow R}(Z) \]  \hspace{1cm} (13)
\[ f_{Q \rightarrow R}(Z) = (Z^{R} - Z^{I})^2 + (Z^{I} - Z^{K})^2 \]  \hspace{1cm} (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,
\[ \Delta W_j = -\alpha \frac{\partial E}{\partial W_j} = -i\alpha \frac{\partial E}{\partial W_j^{R}} - j\alpha \frac{\partial E}{\partial W_j^{I}} - k\alpha \frac{\partial E}{\partial W_j^{K}} \]  \hspace{1cm} (15)

\[ \Delta a_{ji} = -\beta \frac{\partial E}{\partial a_{ji}} = -i\beta \frac{\partial E}{\partial a_{ji}^{R}} - j\beta \frac{\partial E}{\partial a_{ji}^{I}} - k\beta \frac{\partial E}{\partial a_{ji}^{K}} \]  \hspace{1cm} (16)

\[ \Delta b_{ji} = -\gamma \frac{\partial E}{\partial b_{ji}} = -i\gamma \frac{\partial E}{\partial b_{ji}^{R}} - j\gamma \frac{\partial E}{\partial b_{ji}^{I}} - k\gamma \frac{\partial E}{\partial b_{ji}^{K}} \]  \hspace{1cm} (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).
\[ \frac{\partial E}{\partial W_j^{R}} = \frac{\partial}{\partial W_j^{R}} \left( \frac{\partial y}{\partial a_{ji}^{R}} \frac{\partial a_{ji}^{R}}{\partial W_j^{R}} + \frac{\partial y}{\partial a_{ji}^{I}} \frac{\partial a_{ji}^{I}}{\partial W_j^{R}} + \frac{\partial y}{\partial a_{ji}^{J}} \frac{\partial a_{ji}^{J}}{\partial W_j^{R}} + \frac{\partial y}{\partial a_{ji}^{K}} \frac{\partial a_{ji}^{K}}{\partial W_j^{R}} \right) \]  \hspace{1cm} (18)

\[ \frac{\partial E}{\partial a_{ji}^{R}} = \frac{\partial}{\partial a_{ji}^{R}} \left( \frac{\partial y}{\partial a_{ji}^{R}} \frac{\partial a_{ji}^{R}}{\partial W_j^{R}} + \frac{\partial y}{\partial a_{ji}^{I}} \frac{\partial a_{ji}^{I}}{\partial W_j^{R}} + \frac{\partial y}{\partial a_{ji}^{J}} \frac{\partial a_{ji}^{J}}{\partial W_j^{R}} + \frac{\partial y}{\partial a_{ji}^{K}} \frac{\partial a_{ji}^{K}}{\partial W_j^{R}} \right) \]  \hspace{1cm} (19)
Table 2. QNF

<table>
<thead>
<tr>
<th>δ</th>
<th>Evaluation</th>
<th>(w_1), (w_2), (w_3), (w_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0036</td>
<td></td>
<td>(-1, 0.12)</td>
</tr>
<tr>
<td>0.3235</td>
<td></td>
<td>(0.5, 0.12) (0.6, 0.5, 0.4, 0.3)</td>
</tr>
<tr>
<td>0.0008</td>
<td>0.0003</td>
<td>(-0.5, 0.12) (0.6, 0.5, 0.4, 0.3)</td>
</tr>
<tr>
<td>0.4889</td>
<td>0.4889</td>
<td>(0.0, 0.12) (0.6, 0.5, 0.4, 0.3)</td>
</tr>
<tr>
<td>0.0030</td>
<td></td>
<td>(0.0, 0.12) (0.6, 0.5, 0.4, 0.3)</td>
</tr>
<tr>
<td>0.0032</td>
<td></td>
<td>(0.0, 0.12) (0.6, 0.5, 0.4, 0.3)</td>
</tr>
<tr>
<td>0.0030</td>
<td></td>
<td>(1, 0.12) (0.6, 0.5, 0.4, 0.3)</td>
</tr>
</tbody>
</table>

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 rules is twenty-five. However, in the new method, we give \(x_1, x_2, x_3, x_4\) 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).

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 \(\delta\). The results are shown in Tables 3 and 4. Results shown are the average of 20 trials. In these Tables,

\[ E_{all} = \frac{1}{2N}\sum_{d=1}^{N}(Y_d - T_d)^2 \]  

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

\[ x_1, x_2, x_3, x_4 \in \{-0.9, 0, 0.9\} \]  

(34)

Equivalent-625

\[ x_1, x_2, x_3, x_4 \in \{-0.9, -0.5, 0, 0.5, 0.9\} \]  

(35)

Table 3. NF vs. QNF for Function 1

<table>
<thead>
<tr>
<th>Function</th>
<th>NF</th>
<th>(a=0.1), (\beta=0.01), (\gamma=0.001)</th>
<th>QNF</th>
<th>(a=0.1), (\beta=0.01), (\gamma=0.001)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of data</td>
<td>(\delta)</td>
<td>Epoch</td>
<td>Evaluation</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>Random-81</td>
<td>0.003</td>
<td>3519</td>
<td>227</td>
<td>0.0229</td>
</tr>
<tr>
<td>Equivalent-81</td>
<td>0.003</td>
<td>5221</td>
<td>173</td>
<td>0.0259</td>
</tr>
<tr>
<td>Random-625</td>
<td>0.003</td>
<td>3759</td>
<td>297</td>
<td>0.0085</td>
</tr>
<tr>
<td>Equivalent-625</td>
<td>0.003</td>
<td>1663</td>
<td>215</td>
<td>0.0065</td>
</tr>
</tbody>
</table>

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 \(x_1, x_2, x_3, x_4\)) 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

<table>
<thead>
<tr>
<th>Function 2</th>
<th>NF: α=0.1, β=0.01, γ=0.01</th>
<th>QNF: α=0.1, β=0.01, γ=0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of data</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>Random-81</td>
<td>5486</td>
<td>500</td>
</tr>
<tr>
<td>Equivalent-81</td>
<td>5999</td>
<td>322</td>
</tr>
<tr>
<td>Random-625</td>
<td>1125</td>
<td>147</td>
</tr>
<tr>
<td>Equivalent-625</td>
<td>1135</td>
<td>232</td>
</tr>
</tbody>
</table>

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 \(x_1, x_2, x_3, x_4\) 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.](image)

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