Identification of fuzzy neural networks by forward recursive input-output clustering and accurate similarity analysis

DOI:
10.1016/j.asoc.2016.08.009

Citation for published version (APA):

Published in:
Applied Soft Computing Journal

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester’s Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.
Identification of Fuzzy Neural Networks by Forward Recursive Input-output Clustering and Accurate Similarity Analysis

Junfei Qiao 1, 2, Wei Li 1, 2, Xiao-Jun Zeng 3, Honggui Han 1, 2

1. College of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China.
2. Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing 100124, China.
3. School of Computer Science, University of Manchester, Manchester M13 9PL, UK.

Abstract. This paper proposes a two-phase identification approach to Mamdani fuzzy neural networks. The first phase is the system identification which includes a novel forward recursive input-output clustering method for the structure initialization and the gradient descent algorithm for the parameter initialization. The main advantage of the proposed method is that it fits perfectly the special clustering requirement for system identification: coarser clustering in the regions where the identified system is smoother and finer clustering in the regions where the system is more variable or nonlinear. The second phase is the system simplification which includes the accurate similarity analysis and merging method for similar fuzzy rules and the gradient descent algorithm for the parameter finalization. The accurate similarity analysis developed solves the long standing open problem how to compute the exact (rather than approximate) similarity between fuzzy sets and rules with Gaussian membership functions. Numerical experiments based on well-known benchmark data sets are used to verify the effectiveness and accuracy of the proposed approach.

Keywords: System identification, Mamdani fuzzy neural networks, forward recursive input-output clustering, similarity analysis.

1. Introduction

Since the original research by Takagi and Sugeno [1] and Pedrycz [2], the identification and learning of fuzzy systems (FSs) or fuzzy neural networks (FNNs) from data has been one of the most active research areas for three decades, and a number of different approaches have been proposed. Among these, the clustering based approach has attracted most attentions due to its conceptual simplicity, computing effectiveness, and high accuracy.

Originally proposed by Yager and Filev [3] and Chiu [4], the basic idea of the clustering based approach includes two basic steps: The first step is the structure initialization or identification in which a clustering algorithm is used to partition the data space into several clusters and then the identified clusters are converted into the fuzzy rules or neurons to form the initial FSs or FNNs; the second step is that the parameter identification in which an optimization algorithm such as the gradient descent algorithm is used to find the best system parameters.

When applying the above clustering based identification or learning of FSs or FNNs, many different clustering algorithms have been used or proposed (please see [5-6] and [7] for the detailed list). Based on these clustering algorithms, two types of approaches, one stage clustering and two stage clustering, have been proposed in the structure initialization or identification step.

Suppose that the input-output sample data of a given system are given as \{x(t), y(t)\} \(t=1, 2, \ldots, T\), where \(x(t)\) is the input vector at \(t\) with \(x(t)=\{x_1(t), x_2(t), \ldots, x_n(t)\}\) and \(y(t)\) is the corresponding output at time \(t\). In the one stage clustering approach, there are three different methods dependent on what and how the training data are used in clustering. The first method is the input only clustering such as [8-10] and [11] in which only input training data, \{x(t)\} \(t=1, 2, \ldots, T\), are clustered. As a result, the useful information and knowledge related to identified systems from the output data is not
used. This sometimes leads to the ineffective clustering in the sense that there are too few clusters (i.e., underfitting) in the regions where the output data of the identified system are highly variable or nonlinear and too many clusters (i.e., overfitting) in the regions where the output data of the identified system are smooth. The second method is a combined input-output clustering such as [2, 4], [12-17] in which the input-output training data, \( \{x(t), y(t)\} \) \( (t=1, 2, \ldots, T) \) and/or their linear combinations (for T-S fuzzy systems), are clustered. The third method is a weighted input-output clustering such as [5] and [6] in which a weighting factor is added into the objective function of clustering. This basically is equivalent to clustering the weighted input-output training data, \( \{x(t), \lambda y(t)\} \) \( (t=1, 2, \ldots, T) \), are clustered with \( \lambda \) being the weighting factor. The second method is the special case of the third method when \( \lambda=1 \) and thus the third method is a generalization of the second method. In comparing with the input only clustering method, the main advantage of these two methods is that the useful output data are used in the clustering. However, in some degree they still suffer a similar ineffective as the input only clustering method. The reason is as follows: In a region where the output data of the identified system are highly variable or nonlinear, more and finer clusters are needed in order to represent this highly variable or complicatedly nonlinear part of the system. In order to achieve this, a larger weighting factor \( \lambda \) is needed in order to achieve a finer partition. However, in a region where the output data of the identified system are smooth, fewer and coarser clusters are sufficient to represent this simpler part of the system. To this purpose, a smaller \( \lambda \) is needed in order to achieve a coarser partition. Therefore a suitable weighting factor is difficult to find sometimes and does not exist for some complicated systems.

![Fig. 1 Special clustering requirement for system identification](image)

In fact, the identification of FSs or FNNs by its nature is a supervised learning problem and therefore the clustering methods for the system identification need to satisfy some special requirement. Such a requirement can be illustrated by Fig. 1. As it can be seen in Fig. 1, in a region where the identified system is smooth, fewer clusters or neurons are sufficient to represent this system in such a region as the system in this region is simple. As a result, only a coarser clustering is needed in order to avoid overfitting. On the other hand, in a region where the output data of the identified system are smooth, fewer and coarser clusters are sufficient to represent this simpler part of the system. To this purpose, a smaller \( \lambda \) is needed in order to achieve a coarser partition. Therefore a suitable weighting factor is difficult to find sometimes and does not exist for some complicated systems.
regions. By recognizing such a special requirement for clustering methods used in system identification, the first motivation of this paper is, by using the output training data as the supervisory signals to detect where the identified system is smooth or highly variable, to develop an alternative and special designed clustering approach to meet such a requirement in order to improve the effectiveness and accuracy of system identification.

Different from the one stage clustering, the two stage clustering approach divides the clustering process into the two separated stages - the input clustering and output clustering. So far there are only a very limited number of papers such as [18] and [19] based on the two stage approach. The basic idea of the existing approach proposed by [18] and [19] is as follows: Firstly group the output data into the different clusters named as the contexts in [18] or the output constraints in [19]; then for each output cluster, collect and cluster the input data whose corresponding output data are within the given output cluster. Conceptually speaking, the two stage approach fits better to the special clustering requirement for system identification due to the reason below: In a region where the identified system is smoother, the output data have less variation and then less output clusters are needed. As a result, the corresponding input clusters will be less and therefore coarser clustering; in a region where the identified system is highly variable or nonlinear, the output data have more variations and then more output clusters are needed. As a result, the corresponding input clusters will be more and therefore finer clustering. Despite this advantage, the existing two stage approach suffers a different drawback: For each output cluster, its corresponding input data set often has very complicated shapes and highly overlaps with the other input sets for other output clusters. As a consequence, such an input data set is very difficult to be clustered and separated from other input data sets. However, such a difficulty occurred is due to their backward feature: the output clustering first and then the corresponding input clustering. By recognizing the reason behind the difficulty, the second motivation of this paper is to develop a forward two stage clustering approach to overcome such a drawback.

When a FS or FNN is identified by using a clustering based structure initialization and a parameter optimization algorithm, the identified model often are unnecessarily too complicated and overfitted with the poor interpretability due to the following reason: Clustering is a local data partition approach. For a model structure initialized by such a local partition approach and then the model parameters are identified by a global optimization algorithm, the resulting model often turns out being too complicated with various redundant or similar rules or neurons due to the better parameters identified by the global optimization algorithm. To overcome such a drawback, many simplification methods have been proposed such as [9] and [20-27,28]. In all these methods, the key issue is how to identify the highly similar fuzzy rules or neurons to merge. In other words, the key problem to be solved in the model simplification is the similarity analysis and calculation between membership functions (MFs) and between fuzzy rules or neurons. However, for fuzzy sets and fuzzy rules with the most widely used Gaussian MFs, this has been a long standing unsolved problem for almost two decades, because computing the similarity “is very complex because of the nonlinear shape of Gaussian functions” [9,26]. For this reason, all existing solutions for the similarity analysis of Gaussian MFs are based on different simplified, approximation, or heuristic methods rather than accurate similar analysis. For example, in [9,20,21], and [26], triangular or trapezoidal MFs are used to approximate Gaussian MFs and then their similarity is used to replace the similarity between Gaussian functions, whereas some greatly simplified heuristic methods are used in [22-24]. Further, the existing definition and analysis of the similarity between fuzzy rules is also suffering some inaccuracy (see the detailed analysis and example in Section 5). For this reason, there is a particular need and opportunity to develop system simplification methods based on accurate similarity analysis between fuzzy sets and fuzzy rules with Gaussian MFs and this is the third motivation of this paper.
This paper investigates the identification problem of Mamdani FNNs and proposes a new learning approach which includes a two stage forward recursive input-output clustering (FRIOC) method for structure initialization and a model simplification method based on the accurate similarity analysis and computing. Comparing with the existing clustering based approaches, the main features and advantages of the proposed approach can be summarized as below:

- Firstly it meets perfectly to the special clustering requirement for system identification in the sense that coarser clustering in smooth regions and finer clustering in highly variable regions; secondly this is a forward two-stage clustering in the sense that it applies the input clustering firstly and then the output clustering and validity check. Such a forward approach enables the automatic identification of the right number of clusters and avoids the trial-error process and complexity of the existing two stage backward clustering approaches. Therefore it is conceptually simpler and computationally more effective.

- Different from the existing similarity analysis and computing methods where triangular or trapezoidal MFs, or heuristic methods are used to approximate the original Gaussian MFs and their similarity, the proposed approach develops the accurate similarity analysis and computing based on the original Gaussian MFs. More importantly, by analysing and identifying the weakness in the heuristic definition of similarity degree between fuzzy rules in the literature, the similarity degree between fuzzy rules in the proposed approach is defined by following strictly the original definition of the similarity degree between fuzzy sets and calculated accurately by proposing a new computing algorithm. In this way, the long standing weakness by using an approximate similarity analysis in system identification is overcome, and the proposed approach makes it possible for the first time to analyse and compute the similarity degree accurately for fuzzy sets and fuzzy rules with Gaussian MFs.

The paper is organized as follows: Section 2 introduces the Mamdani FNN and its learning problem. Section 3 gives the proposed new framework for the identification of FNNs. Section 4 proposes the system identification method with the FRIOC for the structure initialization. Section 5 develops the system simplification method based on the accurate similarity analysis and merge. Then the simulation examples are given in Section 6 and finally the conclusion is given in Section 7.

2. Mamdani FNNs and Its Learning Problem

2.1 Mamdani FNNs

Mamdani type of FNNs is one of the most widely used FNNs. A basic FNN of Mamdani type consists of five layers, which are input layer, MF layer, fuzzy rule layer, normalized layer and output layer, as shown in Fig. 2.

In this paper, the FNN with multiple inputs and single output (MISO) is considered here. Such a FNN can be represented as follows:

\[
y = FNN(x) = \sum_{j=1}^{r} w_j \bar{u}_j(x)
\]

(1)

where \( y \) is the output of the FNN, \( x \) is the input vector with \( n \) input variables \( x=[x_1, x_2, \ldots, x_n] \), \( n \) is the number of input variables, \( r \) is the number of fuzzy rules, \( w_j \) is the consequent parameter for the \( j \)-th fuzzy rule, \( \bar{u}_j(x) \) is the \( j \)-th normalized layer output of the \( j \)-th fuzzy rule, which is represented by the following:

\[
\bar{u}_j(x) = \frac{u_j(x)}{\sum_{j=1}^{r} u_j(x)} = \frac{\prod_{i=1}^{n} u_{ij}(x_i)}{\sum_{j=1}^{r} \prod_{i=1}^{n} u_{ij}(x_i)}
\]

(2)
where $u_j(x)$ is the output of $j$-th fuzzy rule, and $u_i(x_i)$ is the MF of the $i$-th input in the $j$-th fuzzy rule. In this paper, the Gaussian MF as follows is chosen:

$$u_i(x_i) = e^{-\frac{(x_i-c_{ij})^2}{\sigma_{ij}^2}}$$

where $c_{ij}$ is the center of Gaussian MF for the $i$-th input in the $j$-th fuzzy rule, and $\sigma_{ij}$ is the corresponding width parameter of Gaussian MF.

2.2 Learning problem statement

Suppose that the input-output sample data of a given system are given as $\{x(t), y(t)\}, (t=1, 2, \ldots, T)$, where $x(t)$ is the input vector at $t$ with $x(t)=[x_1(t), x_2(t), \ldots, x_n(t)]$ and $y(t)$ is the corresponding output at time $t$. The problem to be solved is to find a Mamdani FNN which is represented by equation (1) to minimize the model prediction or testing errors with the given system in the Root Mean Square Error (RMSE) sense.

For the above given learning or identification of Mamdani FNN problem, a new framework and approach is going to be proposed in the next section and its corresponding algorithms are then detailed in Sections 4 and 5.

3. New framework and approach for the identification of Mamdani FNNs

For the identification or learning problem of Mamdani FNNs stated in the last section, the new framework and approach proposed in this paper, as illustrated in Fig. 3, includes the two main phases: The first phase is the structure and parameter initialization and the second phase is the structure simplification and parameter finalization.

The first phase includes the two main stages: The first stage is the structure initialization in which a FRIOC method is proposed to initialize the structure of a Mamdani FNN, including the number of fuzzy rules and the parameters of Gaussian functions; the second stage is the parameter initialization in which the gradient descent method is used to optimize all the parameters of the Mamdani FNN based on the initialized structure in the first step. At the end of the first phase, an initial Mamdani FNN is obtained.
As the initialized FNN obtained in the first phase often may not be the most accurate and simplest one, the second phase is designed to simplify and improve the initialized FNN and it includes the three main stages: The first stage is to identify and remove the redundant fuzzy rules or neurons in the initialized FNN, where a fuzzy rule or neuron is regarded as redundant if it makes little contribution to the output of the FNN. The second stage is to identify highly similar fuzzy rules or neurons and then merge them. These first two stages belong to the structure simplification in order to avoid the overfitting, reduce the number of neurons and complexity of the FNN, and improve the consistency and interpretability. After the structure simplification, the parameters of the simplified FNN need to be optimized and therefore the gradient descent optimization method is used to finalize the parameters of the FNN.

Based on the above proposed framework and approach, the detailed methods and algorithms for each of the above phases is going to be developed and presented in the following sections.

4. Structure and parameter initialization for a Mamdani FNN

4.1 Basic idea of forward recursive input-output clustering (FRIOC) approach

In this section, a FRIOC approach which meets the special clustering requirement for system identification described in the introduction is proposed to initialize the structure of a Mamdani FNN. The proposed approach consists of two main stages. The first stage is to partition input data coarsely by applying a k-means clustering method. The second stage is to perform the recursive clustering by checking whether the refined sub-clustering is needed for each cluster by considering the
corresponding output variation and carrying out sub-clustering when needed. If the variation of the corresponding output data within a cluster meets the required accuracy threshold, the cluster does not need to be further sub-clustered, as the corresponding neuron formed by the cluster will be adequate to accurately represent the system in the given cluster. If the output variation within a cluster does not meet the required accuracy threshold, the cluster needs to be further sub-clustered, as the corresponding neuron formed by the cluster will not be sufficient to accurately represent the system in the given cluster. Such a process is applied recursively till the output variation for each cluster is within the acceptable level. This proposed clustering approach fits perfectly the clustering requirement for system identification as the coarser clustering in more smooth regions and finer clustering in more variable regions.

In order to clearly illustrate the proposed clustering approach, firstly some notations are introduced. For the given training data set \( \{ \mathbf{x}(t), \mathbf{y}(t) \} \), \( t = 1, 2, \ldots, T \), where \( \mathbf{x}(t) \) is the input vector at \( t \) with \( n \) input variables \( \{ x_1(t), x_2(t), \ldots, x_n(t) \} \) and \( \mathbf{y}(t) \) is the output at time \( t \), \( m \) and \( M \) are the minimum and maximum values of the output data respectively, that is,

\[
\min_{t=1,2,\ldots,T} \{ y(t) \} = m \quad \text{and} \quad \max_{t=1,2,\ldots,T} \{ y(t) \} = M.
\]

(4)

Suppose the desired output variation or threshold of accuracy is \( \varepsilon (\varepsilon > 0) \) which is a small real value and needs to be defined by a designer according to a considered problem and requirement. The \( K \) is the number of clusters for k-means method in the initial clustering and is defined as the smallest integer such that

\[
\frac{M - m}{K} < \varepsilon.
\]

(5)

In other words, \( K \) is the smallest integer which divides the output data \( [m, M] \) into \( K \) even output intervals such that the length or variation of each output interval is smaller than \( \varepsilon \). That is, \( K \) can be calculated as

\[
K = \left\lceil \frac{M - m}{\varepsilon} \right\rceil,
\]

(6)

where \( \lceil \cdot \rceil \) is the smallest integer function.

Following the basic idea and notations introduced above, in the next subsection, the FRIOC approach is going to be proposed and described in details.

### 4.2 FRIOC approach

At the first stage of the proposed clustering method, a k-means clustering method is used to partition the input data. When applying k-means algorithm to clustering input data, the number of clusters \( K \) can be obtained according to the (6) based on the accuracy value \( \varepsilon \). Suppose the outcome of this first stage input clustering is \( K \) clusters as \( Ic_1, Ic_2, \ldots, Ic_K \).

The initial input clustering only partitions the input training data based on their similarities. However, some initially identified coarse clusters may not be suitable for the construction of the neuron, as the corresponding output data of each of such clusters are not at the same output level. Therefore, the second stage of the proposed clustering method is to decide whether further clustering for each input cluster is needed by checking whether the variation of its corresponding output data meets the required threshold of accuracy (i.e., \( \varepsilon \)). If the output variation for a given input cluster is larger than the threshold, this input cluster needs to be further clustered. The desired clustering outcome such that the output variation of each input cluster is within the given threshold will be obtained by applying this clustering process recursively, which is specified as the below.

In the second stage, firstly for each cluster \( Ic_i \) \( (i = 1, 2, \ldots, K) \) obtained by the k-means clustering method in the first stage, collect all the corresponding output data which is defined as \( Oc_i \) as follows,
\[ OC_i = \{ y(t) \mid x(t) \in I_{c_i} \} \tag{7} \]

Then suppose that \( m_i \) and \( M_i \) are the minimum and maximum values of each output data set \( Oc_i \), respectively, that is,
\[ m_i = \min_{y(t) \in Oc_i} \{ y(t) \}, \quad M_i = \max_{y(t) \in Oc_i} \{ y(t) \}. \tag{8} \]

Now check, for each input cluster \( I_{c_i} \) (\( i=1, 2, \ldots, K \)), whether the further clustering is needed according to its corresponding output variation which is defined as \( M_i - m_i \). If \( M_i - m_i < \varepsilon \), that is, the variation of output data \( Oc_i \) is smaller than the given threshold, then cluster \( I_{c_i} \) does not need to perform input sub-clustering, as this cluster is sufficient to represent this part of the system. If \( M_i - m_i > \varepsilon \), that is, the variation of output data \( Oc_i \) is larger than the given threshold, then cluster \( I_{c_i} \) is not sufficient to represent this part of the system and needs to be further clustered.

Suppose that \( I_{c_i} \) is an input cluster whose output variation is larger than the given and is needed to be further clustered. Let \( K_{s_i} \) be the smallest integer such that
\[ \frac{M_i - m_i}{K_{s_i}} < \varepsilon. \tag{9} \]

Then it can be calculated as
\[ K_{s_i} = \left\lceil \frac{M_i - m_i}{\varepsilon} \right\rceil. \tag{10} \]

Such a \( K_{s_i} \) is used as the cluster number for the further clustering of cluster \( I_{c_i} \). After using the \( k \)-means method to \( I_{c_i} \), the obtained \( K_{s_i} \) sub-clusters are defined as \( I_{c_i1}, I_{c_i2}, \ldots, I_{c_iK_{s_i}} \). Then we need to check whether each \( I_{c_{ij}} \) \((l=1, 2, \ldots, K_{s_i})\) needs to be further sub-clustered using the same method described above. If the variation of output data \( Oc_{ij} \) is smaller than the given threshold of accuracy, then cluster \( I_{c_{ij}} \) does not need to be further sub-clustered. If the variation of output data \( Oc_{ij} \) is larger than the given threshold, input cluster \( I_{c_{ij}} \) needs to be further sub-clustered. Based on the above description, it is clear that the proposed approach is a recursive process, and such a recursive process will end after a finite number of iterations to achieve the input-output clustering goal, that is, for each obtained cluster, the variation of its corresponding output data is within a given.

According to the detailed description as above, there are three main features for the proposed FRIIOC approach. Firstly, the proposed approach fits very well to the special clustering requirement for system identification described in Section 1: coarser clustering in smoother regions and finer clustering in more variable or nonlinear regions. Secondly, this approach uses the output variation in each cluster as the supervisory signal to guide the recursive clustering process and can be regarded as a supervised clustering method as it considers and uses both the input and output information in the clustering process. Thirdly, this approach is able to systematically identify the right number of clusters and avoid the trial-error process and complexity in the existing approaches.

**Remark 1.** For a given system identification or learning problem, there is usually a desired error bound or termination condition such as the RMSE is less than \( E_{\max} \). Then the threshold value of accuracy should be chosen as \( \varepsilon = 2E_{\max} \). In this way, for input cluster \( I_{c_j} \), if its corresponding minimum and maximum output values, \( m_j \) and \( M_j \), satisfy \( M_j - m_j < \varepsilon \), then by assigning the corresponding consequent parameter for cluster \( I_{c_j} \) as \( w_j = 0.5(M_j + m_j) \), the approximation error at each input \( x(t) \in I_{c_j} \) roughly satisfy
\[ |y(t) - FNN[x(t)]| \leq \max[|M_j - FNN[x(t)]|, |m_j - FNN[x(t)]|] \approx \max[|M_j - w_j|, |m_j - w_j|] = 0.5(M_j - m_j) < 0.5\varepsilon = E_{\max} \tag{11} \]

which implied \( \text{RMSE}_j < E_{\max} \) in this cluster. If \( M_j - m_j < \varepsilon \) for all clusters \( I_{c_j} \), \((j = 1, 2, \ldots, K_{C})\), we
have that RMSE_j < E_max for all j = 1, 2, ..., K_c hold and then the RMSE for all training data is less than E_max. Therefore the desired error bound or termination condition is satisfied. This is basically the reason why the goal of recursive clustering is to ensure the output variation at each cluster satisfying M_j - m_j < ε and why ε is called the threshold value of accuracy.

**Remark 2.** Fuzzy c-means is an algorithm more often used in the clustering based system identification in the literature. The main reason using k-means rather than fuzzy c-means in our approach is as follows: As stated in the above, the proposed FRIOC algorithm is an output variation guided clustering approach, where the input space is partitioned recursively along the output-variation reducing direction. For this reason, it requires computing the output variation for each input cluster. As k-means leads to the clusters with hard boundaries, this makes it possible and simple to compute the output variation for each cluster. However, this is not easy to be achieved by using fuzzy c-means, which will produce the clusters with soft boundaries and make it difficult to compute the output variation in each cluster.

### 4.3 Structure initialization algorithm for Mamdani FNNs

When the above FRIOC process is ended, we can apply the resulting clusters for the structure initialization of the FNN. Based on the idea that each cluster forms a neuron or a fuzzy rule, then the number of neurons or fuzzy rules is determined as being equal to the resulting number of the final clusters by the above FRIOC approach and each neuron or fuzzy rule can be constructed from each cluster. The detailed algorithm for such a structure initialization of a Mamdani FNN using the proposed FRIOC approach can be summarized in FRIOC algorithm below.

Once the initial structure of a FNN is obtained by the above steps, the parameter learning of the FNN by a gradient descent algorithm will be used to complete the initialization of FNN.

<table>
<thead>
<tr>
<th><strong>Forward Recursive Input-Output Clustering (FRIOC) Algorithm</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>// Initialization</strong></td>
</tr>
<tr>
<td>( C_{\text{Final}} = () ) // this set is to store the final clusters and is initialized as empty</td>
</tr>
<tr>
<td>( C_{\text{Temp}} = () ), ( C_{\text{ToBeChecked}} = () ) //These sets are to store intermedia clusters and are initialized as empty</td>
</tr>
<tr>
<td>Calculate the initial number of clusters, ( K ), based on (4) and (6)</td>
</tr>
<tr>
<td>Partition the input data using k-means and obtain ( K ) clusters ( C_{\text{ToBeChecked}} = { Ic_1, Ic_2, ..., Ic_K } )</td>
</tr>
<tr>
<td>for ( i = 1 ) to ( K ) do</td>
</tr>
<tr>
<td>For ( Ic_i ), collect all the corresponding output data ( Oc_i ) as (7)</td>
</tr>
<tr>
<td>Calculate the min and max values ( m_i ) and ( M_i ) of output data ( Oc_i ) based on (8)</td>
</tr>
<tr>
<td>if ( M_i - m_i &lt; \varepsilon )</td>
</tr>
<tr>
<td>Add ( Ic_i ) into ( C_{\text{Final}} )</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>Add ( Ic_i ) into ( C_{\text{Temp}} )</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>( C_{\text{ToBeChecked}} = C_{\text{Temp}} )</td>
</tr>
<tr>
<td><strong>//Recursive Iteration</strong></td>
</tr>
<tr>
<td>while ( C_{\text{ToBeChecked}} \neq ) empty</td>
</tr>
<tr>
<td>( L = \text{length of } C_{\text{ToBeChecked}} )</td>
</tr>
<tr>
<td>( C_{\text{Temp}} = () )</td>
</tr>
<tr>
<td>for ( i = 1 ) to ( L ) do</td>
</tr>
<tr>
<td>For ( Ic_i \in C_{\text{ToBeChecked}} ), collect all the corresponding output data ( Oc_i ) as (7)</td>
</tr>
<tr>
<td>Calculate the min and max values ( m_i ) and ( M_i ) of output data ( Oc_i ) based on (8)</td>
</tr>
</tbody>
</table>
\textbf{4.4 Parameter learning using a gradient descent method}

After obtaining the initial structure of Mamdani FNN by the FRIOC approach, the parameters need to be refined and adjusted in the parameter learning process and a gradient descent algorithm is used to update the parameters, which include the center and width parameters of Gaussian MFs and the consequent parameters. The objective function to be minimized is defined as follows:

\[ E = \frac{1}{2}(y_d - y)^2 \]  

where \( y_d \) is the desired output of the system and \( y \) is the neural network output. The goal of the parameter learning is to minimize \( E \).

The consequent parameter \( w_j \) are updated by using the gradient descent method with the momentum item as follows:

\[ w_j(t+1) = w_j(t) - \eta \frac{\partial E}{\partial w_j} + \zeta (w_j(t) - w_j(t-1)) = w_j(t) + \eta (y_d - y)\bar{u}_j + \zeta (w_j(t) - w_j(t-1)) \]  

where \( \eta \) represents the learning rate, \( \zeta \) represents the momentum coefficient, \( \bar{u}_j \) is calculated as (2). Similarly, the center \( c_{ij} \) of Gaussian MF are updated according to (16).

\[ c_{ij}(t+1) = c_{ij}(t) - \eta \frac{\partial E}{\partial c_{ij}} + \zeta (c_{ij}(t) - c_{ij}(t-1)) \]  

where

\[ \frac{\partial E}{\partial y} = y - y_d \quad \frac{\partial y}{\partial w_j} = w_j \quad \frac{\partial \bar{u}_j}{\partial u_j} = \left( \sum_{i=1}^{i'} u_i \right) / \left( \sum_{i=1}^{i'} u_i \right)^2 \quad \frac{\partial u_j}{\partial \mu_j} = \prod_{k=1}^{k} \mu_{kj} \]
Similarly, the width parameter $\sigma_j$ of Gaussian MF can be adjusted in the same way as the center parameter and is not given here for the simplicity.

With the structure initialization based on the proposed FRIOC method and parameter optimization based on the gradient method, an initial FNN is obtained. However, such an initial FNN often turns out to be unnecessarily too complicated for the following reasons: Firstly the FRIOC method is a clustering or local partition based method providing the initial structure complicated enough to represent the considered system but such a structure is not necessarily the simplest and best structure from a global optimization point of view; secondly, the global parameter optimization by the gradient method will move the centers and widths of the initial fuzzy MFs and may create some similar or redundant fuzzy rules or neurons. Therefore in order to obtain an accurate and more compact network and enhance the transparency and interpretability, the identified initial FNN needs to be further simplified and tuned by removing the similar or redundant neurons and this is the topic of the next section.

5. Simplification of initial Mamdani FNNs

As stated before, the initialized FNN often is not the simplest and best possible one as it may be more complicated with more rules than necessary, which leads to overfitting and reducing the interpretability. To overcome such a drawback, a simplification approach is proposed in this section in order to obtain an accurate and more compact network. The proposed simplification approach consists of two stages: The first stage is to identify and remove the redundant neurons or fuzzy rules; the second stage is to accurately calculate the similarity degree between neurons or fuzzy rules and then merge the similar neurons or fuzzy rules. The details will be presented in the following subsections.

5.1 Identification and removing of redundant fuzzy rules

In this subsection, we will first define when a neuron or fuzzy rule is regarded as redundant. Suppose that the given MISO system to be identified is $g(x)=g(x_1, \ldots, x_n)$ defined on the input space $U = \prod_{i=1}^{n} U_i = \prod_{i=1}^{n} [\alpha_i, \beta_i]$ and the initial FNN identified by using the input-output training data for $g(x)$ is given in (1) whose rule base can be represented in the following form:

$$R_j: \quad \text{IF } x_1 \text{ is } u_{1j} \text{ and } \ldots \text{ and } x_n \text{ is } u_{nj}, \text{ THEN } y = w_j \quad j=1, 2, \ldots, r \quad (19)$$

where $u_{ij}(i=1, \ldots, n; j=1, \ldots, r)$ are the MFs given in (3). Rewrite the above rules in the vector form, and then the above rule base can be represented as

$$R_j: \quad \text{IF } x \text{ is } u_j, \text{ THEN } y = w_j \quad j=1, 2, \ldots, r \quad (20)$$

where $x = (x_1, \ldots, x_n)$ and

$$u_j(x) = u_j(x_1, \ldots, x_n) = \prod_{i=1}^{n} u_{ij}(x_i) \quad (j=1, \ldots, r). \quad (21)$$

If one of the above fuzzy rules is never really active in the sense that its active degree for all possible data in the input space $U = \prod_{i=1}^{n} U_i = \prod_{i=1}^{n} [\alpha_i, \beta_i]$ is so small (i.e., near zero or less than a very small positive number) that it is negligible, then such a rule can be regarded as redundant, as it makes little contribution to represent $g(x)$. Formalizing this case, we have the following definition.

**Definition 1.** Let $g(x)=g(x_1, \ldots, x_n)$ be a given system defined on the input space
\[ U = \prod_{i=1}^{n} \prod_{j=1}^{n} [\alpha_i, \beta_j] \] and \( y = \text{FNN}(x) = \text{FNN}(x_1, \ldots, x_n) \) given in (1) be a FNN model for \( g(x) \) with its rule base given in (20). For a pre-given very small positive real number \( \gamma > 0 \), if fuzzy rule in (20) satisfies \( u_j(x) < \gamma \) for all \( x \in U \), then \( R_j \) is called a redundant fuzzy rule for representing \( g(x) \).

**Remark 3.** When \( R_j \) is a redundant fuzzy rule as defined above, that is, \( u_j(x) < \gamma \) for all \( x \in U \), then it can be seen that its corresponding \( j \)-th neuron in (1) makes little contribution to \( \gamma(x) \). In the sense, the \( j \)-th neuron can be regarded as a redundant one. Noticing that the active degree of a fuzzy rule at its centre is 1, this definition implies that a fuzzy rule is redundant if and only if its centre is far outside the input space \( U \) where the FNN is defined and the active degree of the rule at any point inside the input space \( U \) is very close to zero (i.e., less than the very small real number \( \gamma \)).

Based on Definition 1, it can be seen clearly that fuzzy rule \( R_j \) is redundant if and only if
\[
u^\max_j = \max_{x \in U} u_j(x) = \max_{x_i, x_j \in U} \prod_{i=1}^{n} u_{ij}(x_i) = \max_{x_i, x_j \in [\alpha_i, \beta_j]} u_{ij}(x_i) < \gamma.
\]  

(22)

Based on the equation above, it can be seen that, to check whether a fuzzy rule is redundant, the main computing problem needed to solve is to find the following maximal value for each MF
\[
u^\max_j = \max_{x_i, x_j \in [\alpha_i, \beta_j]} u_{ij}(x_i)
\]  

(23)

The following theorem gives a very simple and effective computing solution for this problem.

**Theorem 1.** Let \( u(x) \) defined on interval \( [\alpha, \beta] \) be a Gaussian MF as follows
\[
u(x) = e^{-\frac{(x-c)^2}{\sigma^2}}.
\]  

(24)

Then
\[
u^\max = \max_{x \in [\alpha, \beta]} u(x) = \begin{cases} u(\alpha) & c < \alpha \\ 1 & \alpha \leq c \leq \beta \\ u(\beta) & c > \beta \end{cases}
\]  

(25)

**Proof.** When \( \alpha < c \leq \beta \), for any \( x \in [\alpha, \beta] \) and \( x \neq c \), \( u(x) < 1 = u(c) \) which implies \( u^\max = \max_{x \in [\alpha, \beta]} u(x) = u(c) = 1 \). Now consider the case \( c < \alpha \). In this case, for any \( x \in [\alpha, \beta] \), \( x \geq \alpha > c \) and then
\[
\frac{du(x)}{dx} = -\frac{2}{\sigma^2} e^{-\frac{(x-c)^2}{\sigma^2}} (x-c) < 0
\]

which implies that \( u(x) \) is strictly monotone decrease function on \( [\alpha, \beta] \). Therefore \( u(x) \) achieves its maximal value at \( x = \alpha \). In other words, \( u^\max = \max_{x \in [\alpha, \beta]} u(x) = u(\alpha) \). The case where \( c > \beta \) can be proved similarly and this ends the proof.

Theorem 1 provides a very simple method to calculate the maximal value of a Gaussian MF. Then based on (22) and Definition 1, a redundant rule can be identified very simply and removed if found. This basically forms the first stage in the simplification of initial FNN. After that, the second stage in the model simplification is to identify the similar rules and then merge the found similar rules into the new rules. This second stage is to be discussed in details in the next two subsections.

### 5.2 Similarity analysis and computing between fuzzy rules

In order to identify and merge the similar rules, we first need to develop a method to compute the similarity degree between fuzzy rules. Suppose that the fuzzy rule base is given in (20), then the similarity analysis and computing is to develop a method to identify and calculate the similarity degree between each pair of fuzzy rules \( R_j \) and \( R_k \) \((j \neq k, j, k=1,2,\ldots,r)\), that is, the similarity degree between fuzzy sets \( u_j \) and \( u_k \) which have the MFs \( u_j(x) = \prod_{i=1}^{n} u_{ij}(x_i) \) and \( u_k(x) = \prod_{i=1}^{n} u_{ik}(x_i) \).
There are many existing methods such as [9] and [20-27, 28] for the analysis and computing of the similarity degree between a pair of fuzzy sets \( u_i \) and \( u_k \) or fuzzy rules \( R_j \) and \( R_k \), and their main steps and features are summarized and analyzed as follows:

- The first step is, for each input variable \( x_i \) (\( i=1,2,\ldots,n \)), calculating the similarity degree between fuzzy sets \( u_{ij} \) and \( u_{ik} \) which have Gaussian MFs \( u_{ij}(x_i) \) and \( u_{ik}(x_i) \). Due to the difficulty that computing the similarity degree between fuzzy sets with Gaussian MFs “is very complex because of the nonlinear shape of Gaussian functions” [9] and [26], all existing methods, as pointed out in Section 1, have utilized some approximation or heuristic simplification rather than the exact and accurate calculation based on the original Gaussian MFs.

- The second step is, for each pair of fuzzy rules \( R_j \) and \( R_k \), calculating the similarity degree between fuzzy sets \( u_j \) and \( u_k \). In almost all existing methods, the similarity degree between fuzzy sets \( u_j \) and \( u_k \) or fuzzy rules \( R_j \) and \( R_k \) is defined and calculated as

\[
S_{\text{min}}(u_j, u_k) = \min_{i, j} \left\{ S(u_{ij}, u_{ik}) \right\}
\]

where \( S(u_{ij}, u_{ik}) \) is the similarity degree between \( u_{ij} \) and \( u_{ik} \).

Although used in the existing methods for the simplification of FSs or FNNs such as [9,20,21,26], the above methods have the following two fundamental weaknesses:

- The calculation of similarity degree between fuzzy sets \( u_{ij} \) and \( u_{ik} \) is inaccurate due to using approximated MFs such as triangular or trapezoidal functions rather than original Gaussian MFs.

- The similarity degree definition between fuzzy rules given in (26) is only a heuristic one. From a theoretical point of view, the similarity degree given in (26) does not satisfy and is inconsistent to the original definition of similarity degree which is given below in (27). From an application point of view, this similarity degree often cannot distinguish some obvious difference in the similarity degree between fuzzy rules. For example, consider a three input case with two scenarios. In scenario 1, the similarity degrees between \( u_{ij} \) and \( u_{ik} \) (\( i=1,2,3 \)) are \( S(u_{ij}, u_{ik})=0.3 \) (\( i=1,2,3 \)). Then based on (26), the similarity degree between \( u_j \) and \( u_k \) is \( S_{\text{min}}(u_j, u_k)=0.3 \); in scenario 2, the similarity degrees between \( u_{ij} \) and \( u_{ik} \) (\( i=1,2,3 \)) are \( S(u_{ij}, u_{ik})=0.3 \), \( S(u_{ij}, u_{ik})=0.95 \), and \( S(u_{ij}, u_{ik})=0.95 \) respectively. Then the similarity degree \( u_j \) and \( u_k \) is \( S_{\text{min}}(u_j, u_k)=0.3 \) which is the same as the one in scenario 1. However, it is obvious that \( u_j \) and \( u_k \) in scenario 2 is much more similar than \( u_j \) and \( u_k \) in scenario 1 due to the much higher similarity degree in \( u_{ij} \) and \( u_{ik} \) (\( i=2,3 \)), but unfortunately the similarity degree defined in (26) cannot distinguish such an obvious difference. In other words, the similarity degree given in (26) does not really meet our expectation and is inaccurate in the similarity analysis from an application point of view.

To overcome the above weaknesses in the similarity analysis and computing between fuzzy rules, we propose in this paper to use our recent obtained method [29] for the accurate analysis and computing of the similarity degree between fuzzy rules. For this purpose, the formal definition of the similarity degree between two fuzzy sets is introduced first and then the main result in [29] is introduced.

**Definition 2.** Let \( A \) and \( B \) be fuzzy sets in the universe of discourse \( U \), the similarity degree between \( A \) and \( B \) is

\[
S(A, B) = \frac{M(A \cap B)}{M(A \cup B)}
\]  

(27)

where \( A \cap B \) and \( A \cup B \) denote the intersection and union of \( A \) and \( B \) respectively, and \( M(\cdot) \) is the size of a fuzzy set. That is, if \( C \) is a fuzzy set in \( U \) with its MF as \( u_c(x) \), then \( M(C) \) is defined as

\[
M(C) = \int_U u_c(x)dx.
\]

**Remark 4.** When fuzzy sets \( A \) and \( B \) given above are chosen to be \( u_{ij} \) and \( u_{ik} \) with their MFs given in (3), formula (27) defines the similarity degree between two fuzzy sets with Gaussian MFs; when \( A \)
and $B$ given in the above definition are chosen to be multidimensional fuzzy sets $u_j$ and $u_k$ with their MFs given in (21), then formula (27) defines the similarity degree between two fuzzy rules. From the above definition, it is implied immediately that the similarity degree is symmetry.

Before introducing the main theorem in [29] for calculating the similarity degree between a pair of fuzzy rules, we use Fig. 4 to illustrate the intuitive idea behind the theorem whose proof is rather complicated as shown in [29]. From Fig. 4 it can be seen clearly that the area between $M(A \cup B)$ and $M(A \cap B)$ is the grey part, which is denoted as $M(|A-B|)$. Then Fig. 4 indicates the relationship:

$$M(A \cup B) = M(A \cap B) + M(|A-B|).$$  

(29)

This together with $M(A \cup B) = M(A) + M(B) - M(A \cap B)$ implies

$$M(A \cap B) = \frac{1}{2} [M(A) + M(B) - M(|A-B|)]$$  

(30)

$$M(A \cup B) = \frac{1}{2} [M(A) + M(B) + M(|A-B|)]$$  

(31)

![Diagram](image)

Fig. 4. $M(|A-B|)$ is the area or the grey part formed by three parts: $S_1$, $S_2$, and $S_3$. Therefore $M(A \cup B) = M(A \cap B) + M(|A-B|)$ and $M(|A-B|) = A(S_1) + A(S_2) + A(S_3)$.

Therefore, if $M(|A-B|)$ can be calculated, then $M(A \cup B)$ and $M(A \cap B)$ can be calculated based on (30) and (31) and then the similarity degree $S(A, B)$ based on (27). In the below how $M(|A-B|)$ can be calculated is illustrated based on Fig. 4. Support that the jointed points between Gaussian MFs $u_A(x)$ and $u_B(x)$ are $x_{01}$ and $x_{02}$, then from Fig. 4, it can be seen that $M(|A-B|)$ is formed by three parts:

$$S_1 = \{(x, y) \mid x_{01} \leq x \leq y \leq u_B(x)\}$$  

(32)

$$S_2 = \{(x, y) \mid x_{01} \leq x \leq x_{02}, u_B(x) \leq y \leq u_A(x)\}$$  

(33)

$$S_3 = \{(x, y) \mid x \geq x_{02}, u_A(x) \leq y \leq u_B(x)\}$$  

(34)

By denoting $A(S)$ be the area of set $S$, then $M(|A-B|) = A(S_1) + A(S_2) + A(S_3)$. Further based on the geometry meaning of integral, the above equalities imply

$$M(|A-B|) = A(S_1) + A(S_2) + A(S_3)$$

$$= \int_{x_{01}}^{x_{02}} [u_B(x) - u_A(x)] dx + \int_{x_{01}}^{x_{02}} [u_A(x) - u_B(x)] dx + \int_{x_{02}}^{\beta} [u_B(x) - u_A(x)] dx = \int_{x_{01}}^{\beta} [u_B(x) - u_A(x)] dx.$$  

(35)

That is $M(|A-B|)$ can be calculated as the integral of function $[u_B(x)-u_A(x)]$ on $[\alpha, \beta]$. By formalizing and extending this intuitive idea, we have the following theorem whose proof is given in [29].

**Theorem 2.** Suppose that $A$ and $B$ are a pair of fuzzy sets on $U = \bigcup_{i=1}^{n} [\alpha_i, \beta_i]$ with the MFs as
\[ u_A(x) = \prod_{i=1}^{n} u_{A_i}(x_i) \quad \text{and} \quad u_B(x) = \prod_{i=1}^{n} u_{B_i}(x_i). \]  

Then equalities (29)-(31) hold and

\[
S(A,B) = \frac{M(A) + M(B) - M([A - B])}{M(A) + M(B) + M([A - B])}
\]

(36)
in which \(M(A), M(B)\) and \(M([A-B])\) can be calculated by the following formulas respectively

\[
M(A) = \int_{\alpha_{A_1}}^{\beta_{A_1}} \cdots \int_{\alpha_{A_n}}^{\beta_{A_n}} u_A(x_1)dx_1 \cdots dx_n = \prod_{i=1}^{n} \int_{\alpha_{A_i}}^{\beta_{A_i}} u_{A_i}(x_i)dx_i
\]

(37)

\[
M(B) = \int_{\alpha_{B_1}}^{\beta_{B_1}} \cdots \int_{\alpha_{B_n}}^{\beta_{B_n}} u_B(x_1)dx_1 \cdots dx_n = \prod_{i=1}^{n} \int_{\alpha_{B_i}}^{\beta_{B_i}} u_{B_i}(x_i)dx_i
\]

(38)

\[
M([A - B]) = \int_{\alpha_{A_1}}^{\beta_{A_1}} \cdots \int_{\alpha_{A_n}}^{\beta_{A_n}} u_A(x_1)dx_1 \cdots dx_n - \prod_{i=1}^{n} \int_{\alpha_{A_i}}^{\beta_{A_i}} u_{A_i}(x_i)dx_i
\]

(39)

Based on Theorem 2, for a pair of fuzzy rules \(R_j\) and \(R_k\) whose corresponding fuzzy sets are \(u_j\) and \(u_k\) with MFs \(u_j(x) = \prod_{i=1}^{n} u_{j_i}(x_i)\) and \(u_k(x) = \prod_{i=1}^{n} u_{k_i}(x_i)\), the accurate similarity degree between \(u_j\) and \(u_k\) can be obtained by: Step 1. Calculate \(\int_{\alpha_{j_i}}^{\beta_{j_i}} u_{j_i}(x_i)dx_i\), \(\int_{\alpha_{k_i}}^{\beta_{k_i}} u_{k_i}(x_i)dx_i\); Step 2. Calculate \(M(u_j), M(u_k), \) and \(M([u_j - u_k])\) based on (37)-(39); Step 3. Obtain \(S_{j,k} = S(u_j, u_k)\) based on (36). As the calculations in these steps involve only simple integrals and some simple algebraic manipulations, Theorem 2 provides a very simple and effective method to accurately calculating the similarity degree between fuzzy rules.

### 5.3 Identification and merging of similar fuzzy rules

Based on the proposed method of similarity degree calculation between fuzzy rules described in the last subsection, in this subsection, the algorithm to identify and merge highly similar fuzzy rules for the structure simplification is developed step by step as below:

First of all, we calculate the similarity degree for each pair of fuzzy rules based on the method in the last subsection and all these similarity degrees then form the similarity matrix in the following:

\[
S_0 = \begin{bmatrix}
S_{1,2} & \cdots & S_{1,j} & S_{1,j+1} & \cdots & S_{1,k} & S_{1,k+1} & \cdots & S_{1,r} \\
& \cdots & \vdots & & \vdots & \cdots & \vdots & & \vdots \\
S_{j-1,j} & S_{j-1,j+1} & \cdots & S_{j-1,k} & S_{j-1,k+1} & \cdots & S_{j-1,r} \\
& \vdots & \vdots & & \vdots & \cdots & \vdots & & \vdots \\
S_{j-1,k} & S_{j-1,k+1} & \cdots & S_{j,k} & S_{j,k+1} & \cdots & S_{j,r} \\
& \vdots & \vdots & & \vdots & \cdots & \vdots & & \vdots \\
S_{k-1,k} & S_{k-1,k+1} & \cdots & S_{k-1,r} \\
& \vdots & \vdots & & \vdots \\
S_{r-1,r} & & & & & & & & \\
\end{bmatrix}
\]

(40)

\[S_{j,k}\ (j<k, j=1,2,\ldots,r-1; k=2,3,\ldots,r)\] represents the similarity degree between two fuzzy rules \(u_j(x)\) and \(u_k(x)\), and \(r\) is the number of fuzzy rules in the initial FNN. As the similarity degree is symmetry, the above similarity matrix only records the similarity degrees between the \(j\)-th and \(k\)-th fuzzy rules satisfying \(j<k\) for the reason of simplicity and avoiding the redundant or repeat matrix elements.

Secondly we calculate the highest similarity degree within the similarity matrix above, that is,

\[S_{\max} = \max_{\{j,k=1,2,\ldots,r;j<k\}} S_{j,k} = \max_{\{j,k=1,2,\ldots,r;j<k\}} S(u_j, u_k)
\]

(41)

If \(S_{\max}<S_t\), where \(S_t\) is the given threshold of similarity degree, then no pair of fuzzy rules has the
high enough similarity degree and no merger is needed. Hence the merger process is terminated. If $S_{\text{max}} > S_t$, then the similarity degree for at least one pair of fuzzy rules $u_j$ and $u_k$ is high enough and the merger is needed to be applied in order to achieve the desired simplification. Suppose fuzzy rules $u_j$ and $u_k$ are the pair rule with the highest similarity degree which is larger than the pre-defined similarity degree threshold $S_t$, that is, $S_{j,k} = S(u_j, u_k) > S_{\text{max}} > S_t$. Then this pair of rule needs to be merged to form a new rule $G_0$. Assume that the MFs of the antecedent part in this pair of highly similar rules are

$$u_j(x) = \prod_{i=1}^{n} u_j(x_i) = \prod_{i=1}^{n} e^{-\frac{(x_i - c_{j,i})^2}{\sigma_{j,i}^2}} \quad \text{and} \quad u_k(x) = \prod_{i=1}^{n} u_k(x_i) = \prod_{i=1}^{n} e^{-\frac{(x_i - c_{k,i})^2}{\sigma_{k,i}^2}} \quad \text{(42)}$$

Then the first step for the rule merging is to generate the membership functions for the merged new rule $G_0$ and the detailed operations are given as below: The new centre and width parameters for the merged MFs can be obtained by average value of centres and width parameters of the two similar Gaussian MFs given in (42) above, that is

$$c_{i,G_0} = \frac{1}{2} (c_{j,i} + c_{k,i}) \quad \text{and} \quad \sigma_{i,G_0} = \frac{1}{2} (\sigma_{j,i} + \sigma_{k,i}) \quad \text{(43)}$$

where $c_{i,G_0}$ and $\sigma_{i,G_0}$ are the new centre and width parameter of the merged MF $u_{i,G_0}(x_i) (i = 1,...,n)$, respectively. In this way, we obtain the merged new MFs as follows:

$$u_{i,G_0}(x_i) = e^{-\frac{(x_i - c_{i,G_0})^2}{\sigma_{i,G_0}^2}} \quad \text{and} \quad u_{G_0}(x) = \prod_{i=1}^{n} u_{i,G_0}(x_i). \quad \text{(44)}$$

where $u_{G_0}(x)$ forms the merged new MF for the antecedent part of the merged rule $G_0$.

In the second step of rule merging, the consequent parameter of the merged fuzzy rule $G_0$ (i.e., the connection weight of the merged new neuron) needs to be compute. Different from the existing methods [9,20,22-24,26], where the new consequent parameter is determined as the simple average between the two consequent parameters of the original pair of fuzzy rules being merged, a new optimization method as below is proposed to determine the best possible consequent parameter: Suppose that the pair of rules being merged are rules $j$ and $k$ and the index set for this pair of rules is denoted as $M_{ij} = \{j,k\}$. After merging the highest similar rules $j$ and $k$, the resulting FNN takes the form as below

$$FNN_{\text{Merged}}(x) = \sum_{j=1}^{r} \frac{u_j(x)}{\sum_{j \in M_{ij}} u_j(x)} w_j + \frac{u_{G_0}(x)}{\sum_{j \in M_{ij}} u_j(x)} w_{G_0} \quad \text{(45)}$$

in which the consequent parameter $w_{G_0}$ needs to be decided. The proposed method to determine this parameter is to find the best one, $w_{G_0}$ which minimize the mean squares error or RMSE as below

$$w_{G_0}^* = \arg \min_{w_{G_0}} F(w_{G_0}) = \arg \min_{w_{G_0}} \frac{1}{T} \sum_{t=1}^{T} \{y(t) - FNN_{\text{Merged}}[x(t)]\}^2. \quad \text{(46)}$$

The above minimization problem can be solved from $\frac{dF(w_{G_0})}{dw_{G_0}} = 0$ and checking $\frac{d^2 F(w_{G_0})}{dw_{G_0}^2} > 0$, and then the obtained optimal consequent parameter is
where \( \{x(t), y(t)\} \) is the input-output training pair at time \( t \), \( u_j[x(t)] \) can be calculated as (42) and \( u_{G0}[x(t)] \) can be calculated as (44). Once \( w_{G0}^* \) is computed from (47), then the rule merging is completed. The obtained new rule after merging is \( G_0 \), whose MF in the antecedent part is \( u_{G0}(x) \) given in (44) and consequent parameter is \( w_{G0}^* \).

In comparing with the existing methods to determine the merged consequent parameters, there are three main benefits behind our proposed method above:

- The parameter \( w_{G0}^* \) obtained in the proposed method is an optimal one in the sense that leads to the minimal RMSE in comparing with the existing methods.
- The RMSE under the optimal parameter \( w_{G0}^* \) provides the measurement to select the given threshold of similarity degree, which is an issue having not been properly investigated in the literature. For a selected threshold, if the RMSE for the merged FNN under \( w_{G0}^* \) is much worse than the one for the initial FNN, then it means that the selected threshold is too small, which leads to ineffective merger in the sense that such merger or simplification leads to the significant loss of accuracy. As a result, a larger threshold is needed in this case.
- In order to find the best parameters for the final FNN, the merged FNN will need further fine tuning by the gradient descent algorithm, which is sensitive to the initial solution. Therefore the better parameters such as \( w_{G0}^* \) is very useful to enhance the chance to obtain the best possible final FNN when using the gradient descent algorithm in the final parameter tuning.

After identifying the pair of fuzzy rules with the highest similarity (that is, fuzzy rules \( j \) and \( k \)), the merged new fuzzy rule or neuron can be obtained by the above steps and then the merged FNN is identified. However, this merging process does not just stop here, as there may exists some other highly similar fuzzy rules which are worth merging in order to find the simplest and most accurate final FNN. The continuous merging step is as follows: At the beginning, we recalculate the similarity degrees for all rules in the merged FNN. As the similarity degrees between each pair of fuzzy rules have been calculated at the previous steps except those between the merged new rule \( G_0 \) and the unmerged rules \( k_{unmerged}=1, \ldots, j-1, j+1, \ldots, k-1, k+1, \ldots, r \), then only these similarity degrees need to be calculated and then the updated similarity matrix which represents the similarity degrees between all fuzzy rules of the merged FNN are obtained as follows:
Once this new similarity matrix is obtained, the pair of rules with the highest similarity degree is identified and then merged if their similarity degree is over the given threshold. This process is implemented iteratively until the highest similarity degree in the similarity matrix is less than the given threshold and the merger process is terminated. At the end of this process, the structure of the simplified Mamdani FNN is identified and then the final model parameters are obtained by the gradient descent algorithm.

According to the description as above, there are several advantages for the proposed similarity calculation and similarity based merger approach. Firstly, the proposed similarity calculation approach is able to accurately calculate the similarity degree for fuzzy rules with Gaussian MFs rather than approximately or heuristically calculate the similarity degree in the existing methods. Secondly, the proposed similarity based merger approach obtains the optimal consequent parameters for the merged rules rather than the simply average values in the existing methods and more likely leads to the global optimal parameters for the final FNN. Thirdly, the proposed approach addresses the unsolved issue in the literature about how to select and decide the right threshold of the similarity degree for merging. Finally, different from the existing methods [9] and [26] without specifying how to select the right fuzzy rules to merge when there are more a pair of highly similar rules, the proposed approach selects the highest similar pair of rules to merge in the descent order iteratively. Such an approach avoids the possibility that a pair of rules with lower similarity degree is merged first and then causes a pair of rules with higher similarity degree being lost out.

### 5.4 FNN simplification algorithm and the learning procedure

Following the discussion and analysis in the last subsection, the detailed algorithm of the structure simplification for the initial Mamdani FNN can be illustrated by Fig. 5 and summarized as follows:

**Stage 1. Structure simplification algorithm for identifying and removing redundant fuzzy rules:**

**Step 0.** For rule $R_j (j = 1, 2, ..., r)$, calculate $u^\text{max}_j$ defined in (22) based on Theorem 1. If $u^\text{max}_j < \gamma$, then remove rule $R_j$.

**Stage 2. Structure simplification algorithm for identifying and merging highly similar fuzzy rules:**

**Step 1.** Calculate the similarity degree between each pair of fuzzy rules based on formulas (36)-(39) in Theorem 2 and store these similarity degrees in the form of the similarity matrix as (40).

**Step 2.** Determine the highest similarity degree $S_{\text{max}}$ in the similarity matrix, and then check whether the corresponding pair of fuzzy rules $u_j$ and $u_k$ whose similarity degree reaches $S_{\text{max}}$ need to be merged according to the given threshold of the similarity degree $S_r$. If $S_{\text{max}} < S_r$, then no merger is needed and the merger process is terminated. If $S_{\text{max}} > S_r$, then the pair of fuzzy rules $u_j$ and $u_k$ need to be merged to form a new rule and go to step 3.

**Step 3.** Carry out the merger of two fuzzy rules $u_j$ and $u_k$ according to (43) and (47) to obtain the
new center and width parameters (i.e., the antecedent parameters) and the new consequent parameters for the merged new fuzzy rule.

**Step 4.** Calculate the similarity degree based on formula (36-39) in Theorem 2 between the merged new rule and all the remaining rules, respectively, and obtain the updated similarity matrix as (48). Then check whether the highest similarity degree in the updated similarity matrix is higher than the given threshold. If yes, the pair of rules with the highest similarity degree is merged using step 3. This process is implemented iteratively and is terminated until \( S_{\text{max}} < S_t \).

**Step 5.** The simplified structure Mamdani FNN is obtained when the above steps are terminated.

![Diagram](image)

*Fig. 5 Structure simplification procedure for Mamdani FNN*

After the similarity calculation and similarity based merger for highly similar rules, the obtained FNN model is structurally simpler and more interpretable. However, the parameters of the obtained model may not be the best ones to minimize the model errors globally. Therefore in order to improve the accuracy of the simplified model, a gradient descent algorithm is used to finally adjust the model parameters according to (15)-(18) presented in section 4.4. Through this last step of the parameters finely adjusted, the final FNN is obtained with the simplified structure and best accuracy.

### 5.5 Complexity analysis and comparison

As shown in the above two sections, the proposed approach includes two phases: The first phase is the structure and parameter initialization and the second phase is the structure simplification and parameter finalization. In fact, the first phase by itself is a complete identification algorithm for FNNs. The only reason that this algorithm is called structure and parameter initialization is to distinguish a FNN obtained in this phase with the one after the further simplification in the second phase. As it is going to be seen in the simulation examples given in the next section, this algorithm can identify FNNs with high accuracy even without the simplification algorithm in the second phase. For this reason, here we first analyze and compare the complexity of the proposed identification algorithm in Phase 1 with those existing algorithms without the simplification phase.
In our proposed identification algorithm without the simplification phase (i.e., Phase 1 only), firstly, the clustering process for the structure identification starts using all input-output training data to determine the number of the initial clusters based on the output variation threshold or the threshold of accuracy $\epsilon$. Secondly, the clustering iteration starts from the initial clustering for all training data, and follows by the sub-clustering for subsets of all the training data, till all identified clusters satisfy the output variation threshold. At each step of the clustering iteration, those clusters whose output variations are larger than the threshold are further sub-clustering. As each sub-clustering is applied a smaller cluster than the previous iteration, each sub-clustering becomes simpler and less time-consuming than the previous one. At the end of this process, the obtained clusters are used to generate the Gaussian membership functions for the initial FNN. Finally, the gradient descent method is used to determine and tune the parameters of the FNN.

On the other hand, the main steps in the existing clustering based identification algorithms are: Firstly, the clustering process for the structure identification starts by selecting a max number of clusters. Such a max number needs to be large enough in order to ensure that there will be sufficient number of clusters being used to present the system to be identified. As a result, it will be larger than the resulting number of clusters in our algorithm. Secondly, the iteration process of clustering starts from the min number of clusters (often from 2) to the max number. At each step of the iteration, k-means, fuzzy c-means (most often), or another clustering algorithm is applied by using all training data, and at the end of each step a cluster validity index value is calculated. When the iteration process is completed, the clustering result with the best validity index value is selected as the final clusters which are used to generate the Gaussian membership functions for the initial FNN. Finally, a gradient descent method is used to determine and tune the parameters of the FNN.

Based on the above analysis, it can be concluded that our proposed approach is much simpler and more effective than the existing approaches for the two reasons: firstly it requires less iteration in the structure initialization because the existing approaches basically takes an exhausted iteration which carries out all possible clustering steps from the min to max number of clusters, whereas our approach takes an effective iteration which carries out only the necessary clustering steps and automatically identify the required number of clusters. Secondly, in each step of the clustering iteration, the existing approach requires clustering all training data, whereas our approach requires clustering smaller and smaller subsets of all the training data in the iterative process. As going to be seen in our simulation example in the next section, our simpler approach given in Phase 1 often can obtain more accurate results than most existing approaches.

Once the first phase is completed and a FNN is identified or initialized, our approach moves to the second phase to further simplify the obtained FNN. In comparing with the most of the existing approaches where there is no simplification phase, this requires some extra computing steps including identifying and removing redundant rules, computing the accurate similarity between fuzzy rules, and applying the gradient descent algorithm to finalize the parameters of the FNN. Even comparing those approaches with the simplification steps, our approach is still more complicated as the computing of the accurate similarity is more complicated than computing the approximation similarity. By considered this extra phase to simplify and optimize the obtained FNN from the first phase, the overall complexity of our approach is more complicated than the existing approaches due to the extra or more accurate simplification phase. However, such a more complication from an algorithm complexity point of view is a price worth paying, as it gives the simpler and more accurate FNNs.

In fact, our proposed approach includes two separable methods: FNN identification method (i.e., Phase 1) and FNN simplification method (i.e., Phase 2). Therefore, from an application point of view, these methods can be picked and mixed with the existing approaches to meet the need for a particular
application if required. For example, if an application needs an effective method to identify a system but does not require the simplest system model, then the identification method in Phase 1 can be chosen to meet such a need without needing Phase 2. On the other hand, if an application chooses another method for the FNN identification for any reason, then the simplification method in Phase 2 can still be applied to the identified FNN obtained by the chosen method to achieve the model simplification and accuracy improvement.

6. Simulation experiments

In order to illustrate the performance of the proposed approach for the identification of Mamdani FNNs, several benchmark simulation examples such as function approximation and dynamical system identification are presented in this section, and the performance of the proposed approach is compared with the existing methods. All the simulation experiments in this section are implemented by Matlab, and Mean Square Error (MSE) and RMSE given below are utilized as the evaluation indexes:

$$MSE = \frac{1}{T} \sum_{t=0}^{T} (y_g(t) - y(t))^2, \quad RMSE = \sqrt{MSE}$$

where $T$ is the total number of the training data or testing data, $y_g(t)$ is the actual output at time $t$ and $y(t)$ is the FNN output at time $t$. Further, in order to give a fairer measurement of the performances and remove the impact from the randomly generated datasets in the simulation results given in this section, the normalized root mean square error (NRMSE) defined below is also given as an additional evaluation index

$$NRMSE = \frac{RMSE}{\max_{t=1,2,...,T} y_g(t) - \min_{t=1,2,...,T} y_g(t)}$$

NRMSE takes its values between 0 and 1. For an identified FNN, the smaller its NRMSE, the more accurate the FNN. As NRMSE gives a relative (percentage type) error measure independent from the ranges or scales of the datasets, such an additional index gives a more intuitive and fairer measurement of the performances.

Simulation 1. The benchmark dynamical system to be identified is given as follows:

$$y(t+1) = \frac{y(t)y(t-1)[y(t) + 2.5]}{1 + y^2(t) + y^2(t-1)} + u(t)$$

where $u(t) = \sin(2\pi t/25), t \in [1,400]$. In this example, 400 groups of input and output data are generated. The first 200 data are used for training and the rest 200 data for testing. This plant has been modeled by [30]-[34]. The simulation results are shown in Fig. 6 and Table 1.
outputs and the outputs of the identified FNN. It clearly illustrates that the proposed approach performed very well when applying to the identification of the given dynamical system.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of neurons</th>
<th>Number of parameters</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFNN [30]</td>
<td>6</td>
<td>48</td>
<td>0.0283</td>
<td>0.1315</td>
</tr>
<tr>
<td>SFNNEKF [31]</td>
<td>15</td>
<td>105</td>
<td>0.087</td>
<td>--</td>
</tr>
<tr>
<td>GA [32]</td>
<td>7</td>
<td>37</td>
<td>0.0625</td>
<td>0.05</td>
</tr>
<tr>
<td>MCFC [33]</td>
<td>--</td>
<td>--</td>
<td>0.0874</td>
<td>0.0874</td>
</tr>
<tr>
<td>ABC algorithm [34]</td>
<td>10</td>
<td>70</td>
<td>0.2573</td>
<td>0.2088</td>
</tr>
<tr>
<td><strong>Our approach</strong></td>
<td><strong>5</strong></td>
<td><strong>35</strong></td>
<td><strong>0.0481</strong></td>
<td><strong>0.0225</strong></td>
</tr>
</tbody>
</table>

---
The results are not listed in the original papers.

In this example, the final obtained FNN by using the proposed approach requires only 5 fuzzy rules or neurons. Table 1 is the comparison between our proposed approach and several existing methods in the number of neurons and parameters and RMSE. From the table, it is clearly shown that the performances including the number of neurons, the number of parameters, RMSE for testing based on our approach are much better than the other existing methods. From Table 1, firstly, it is clearly shown that, in comparing with the existing methods and results, the FNN obtained by our approach is the most accurate one with the testing RMSE as 0.0225, which is less than half of the testing RMSE 0.05 given in [32] as the most accurate existing result in the literature; secondly, the FNN from our approach is the simplest and most compact one. In comparing with the most accurate existing model given in [32], our FNN with 5 neurons and 35 parameters are significantly less than that one with 7 neurons and 37 parameters. In comparing with the other simplest FNN given in [34], our FNN has the same numbers of neurons and parameters, our testing RMSE as 0.0225 is less than one-seventh of the testing RMSE 0.16 for that FNN obtained in [34]; thirdly, the FNN obtained by our approach has no redundant or highly similar fuzzy rules, which shows that our structure initialization method sometimes can obtain very simple and accurate FNN even without the simplification phase, and therefore confirms the effectiveness of our method; fourthly, the training and testing NRMSEs for the obtained FNN are 0.01 and 0.0047 respectively, which are very close to 0 and therefore further confirm the accuracy of the obtained FNN by the proposed approach. In summary, the identified FNN based on our approach has the best accuracy, the simplest and most compact network structure in compared with the other existing methods. In other words, this example shows the convincing performance of our proposed approach for the identification of dynamical systems.

**Simulation 2.** A function approximation problem where the considered function is

\[
y = 0.6\sin(\pi x) + 0.3\sin(3\pi x) + 0.1\sin(5\pi x)
\]

where \( x \in [-1,1] \). 200 pairs of input and output data within the interval \([-1,1]\) are randomly generated. The first 100 data are used for training and the rest 100 data are used for testing. The simulation results are shown in Table 2.

In this experiment, firstly 12 clusters are obtained in the initialized FNN based on the proposed structure initialization method. Then by applying the structure simplification approach to the initialized FNN, a pair of similar fuzzy rules is identified and then merged, and the obtained final FNN includes 11 fuzzy rules. More importantly, as shown in Table 2, the simplified final FNN are more accurate with the test RMSE 0.0441 than the initialized one with the testing RMSE as 0.0463,
which shows the usefulness of the simplification phase. Secondly, it is clearly shown in Table 2 that, in comparing with the existing methods and results, the FNN obtained by our approach is the most accurate one. Although our model requires a few more parameters than the best existing method in [19], this is a price to pay in order to achieve better accuracy. Thirdly, the training and testing NRMSEs for the FNN before the simplification are 0.0130 and 0.0356 and after the simplification are 0.0114 and 0.0339 respectively, which are all very close to 0 and therefore further confirm the accurate performance by the proposed approach. Besides, the smaller training and testing NRMSEs after the simplification show the improvement in both accuracy and network structure, and then verify the usefulness and effectiveness of the proposed simplification method.

Table 2: Comparisons between different methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of fuzzy rules</th>
<th>Number of parameters</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang et al. [19]</td>
<td>7</td>
<td>21</td>
<td>0.0762</td>
<td>0.1089</td>
</tr>
<tr>
<td>Wang et al. [19]</td>
<td>8</td>
<td>24</td>
<td>0.0241</td>
<td>0.0492</td>
</tr>
<tr>
<td>Pedrycz et al. [18]</td>
<td>10</td>
<td>30</td>
<td>0.1470</td>
<td>0.1620</td>
</tr>
<tr>
<td>Pedrycz et al. [18]</td>
<td>12</td>
<td>36</td>
<td>0.1440</td>
<td>0.1640</td>
</tr>
<tr>
<td>Our approach without simplification</td>
<td>12</td>
<td>36</td>
<td>0.0169</td>
<td>0.0463</td>
</tr>
<tr>
<td><strong>Our approach with simplification</strong></td>
<td><strong>11</strong></td>
<td><strong>33</strong></td>
<td><strong>0.0148</strong></td>
<td><strong>0.0441</strong></td>
</tr>
</tbody>
</table>

Simulation 3. A function approximation problem where the considered function is

\[ y = 1.1(l - x + 2x^2)\exp(-x^2/2) \]  

where \( x \in [-4,4] \). 400 pairs of input and output data within the interval \([-4,4]\) are generated randomly. The first 200 data are used for training and the rest 200 data are used for testing. The simulation results are shown in Table 3.

Table 3: Comparisons between different methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of neurons</th>
<th>Number of parameters</th>
<th>Training RMSE</th>
<th>Testing RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFNN [30]</td>
<td>6</td>
<td>24</td>
<td>0.0056</td>
<td>0.0054</td>
</tr>
<tr>
<td>FAOS-PFNN [35]</td>
<td>5</td>
<td>15</td>
<td>0.0089</td>
<td>--</td>
</tr>
<tr>
<td>OLS [36]</td>
<td>7</td>
<td>22</td>
<td>0.0095</td>
<td>0.0090</td>
</tr>
<tr>
<td>M-RAN [37]</td>
<td>8</td>
<td>25</td>
<td>--</td>
<td>-0.0438</td>
</tr>
<tr>
<td>RANEKF [38]</td>
<td>13</td>
<td>40</td>
<td>0.0262</td>
<td>-0.0262</td>
</tr>
<tr>
<td>ILSFNN [39]</td>
<td>7</td>
<td>21*</td>
<td>--</td>
<td>0.0067</td>
</tr>
<tr>
<td><strong>Our approach</strong></td>
<td><strong>6</strong></td>
<td><strong>18</strong></td>
<td><strong>0.0055</strong></td>
<td><strong>0.0053</strong></td>
</tr>
</tbody>
</table>

-- The results are not listed in the original papers.
* Given 7 rules with three parameters per rule (center, width and consequent parameters), the number of parameters should be 21 rather than 15 listed in [39].

In this example, the obtaining final FNN has 6 fuzzy rules or neurons. Table 3 is the comparison between our approach and several existing methods in the number of neurons and parameters and RMSE of training and testing. The training RMSE for the methods in [30], [35], [38], and [36] are all from [35] and the testing RMSE for the methods in [30], [35], [38], and [36] are all from [30]. From table 3, firstly it can be seen that the final FNN obtained by our approach is the most accurate with the training and testing RMSE as 0.0055 and 0.0053 respectively. Comparing the most accurate existing result in the literature with the training and testing RMSE as 0.0056 and 0.0054 given in [30],...
the FNN obtained by our approach uses 18 parameters in comparing 24 parameters in [30], and therefore is 33% simpler with the better accuracy in comparing with the most accurate existing result in the literature. Secondly, it can be seen that the FNN obtained by our approach is simpler (less number of rules and parameters) than all other methods except the one obtained by FAOS-PFNN [35]. With 18 parameters with the training RMSE as 0.0055 in our FNN in comparing 15 parameters with the training RMSE as 0.0089, our FNN is 16% more complicated than the one by FAOS-PFNN but 38% more accurate and therefore achieves the overall better performance. Thirdly, the training and testing NRMSEs for the obtained FNN are 0.0020 and 0.0019 respectively, which further confirm the high accuracy obtained by the proposed approach.

In conclusion, this example shows that our proposed approach has the best training and testing accuracy and often simpler in comparing with the existing methods.

Simulation 4. The benchmark function approximation problem where the considered function having been modelled by [9], [13], [14], [16], [17], and [40] is

\[ y = f(x_1, x_2) = (1 + x_1^2 + x_2^{1.5})^2 \]  

where \( x_1 \in [1.5], x_2 \in [1.5] \) and 100 groups of input and output data are randomly generated and 50 data are used for training and the remaining 50 data for testing. To compare the performance between our approach and many existing methods, the testing MSE rather than RMSE is used here in order to make the simulation result be consistent with those given in [9], [13], [14], [16], [17], and [40]. The simulation results are shown in Table 4.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of neurons</th>
<th>Number of parameters</th>
<th>Testing MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sugeno et al. [40]</td>
<td>6</td>
<td>65</td>
<td>0.0790</td>
</tr>
<tr>
<td>Kim et al. [13]</td>
<td>3</td>
<td>21</td>
<td>0.0197</td>
</tr>
<tr>
<td>Kim et al. [14]</td>
<td>3</td>
<td>21</td>
<td>0.0090</td>
</tr>
<tr>
<td>Kung et al. [17]</td>
<td>3</td>
<td>21</td>
<td>0.0196</td>
</tr>
<tr>
<td>Chen et al. [9]</td>
<td>4</td>
<td>--</td>
<td>0.0078</td>
</tr>
<tr>
<td>Li et al. [16]</td>
<td>6</td>
<td>42</td>
<td>0.0085</td>
</tr>
<tr>
<td>Our approach without simplification</td>
<td>8</td>
<td>40</td>
<td>0.0114</td>
</tr>
<tr>
<td><strong>Our approach with simplification</strong></td>
<td><strong>6</strong></td>
<td><strong>30</strong></td>
<td><strong>0.0053</strong></td>
</tr>
</tbody>
</table>

-- The results are not listed in the original papers.

In this experiment, firstly 8 clusters are obtained in the initialized FNN by using the proposed structure initialization method. By applying the proposed structure simplification approach to the initialized FNN, one fuzzy rule is identified as redundant (i.e., near zero over the interval of training data) and then removed. Further, a pair of similar fuzzy rules is identified and merged to form a new fuzzy rule. In this way, the initialized FNN with 8 rules or neurons is simplified and a much more compact final FNN with 6 rules or neurons is obtained. More importantly, as shown in Table 4, the simplified final FNN are much more accurate with the test RMSE 0.0053 than the initialized one with the testing RMSE as 0.0114, which shows the great usefulness of the simplification phase. Secondly, it is clearly shown in Table 4 that, in comparing the existing methods and results, the FNN obtained by our approach is the most accurate one, reducing the MSE by 41% in comparing with the best existing result in the literature with MSE as 0.0090 given in [14]. Besides, the obtained FNN also has some good training and testing NRMSEs as 0.0187 and 0.0127 respectively. Although our
model requires a slightly more parameters in a few cases, it is much more accurate in these cases. This again shows convincingly that the proposed approach is a very effective method for function approximation problems.

7. Conclusions

This paper proposes a new learning approach to Mamdani FNNs. The approach is consisted of two phases: The first phase is the system identification which includes the FRIOC method for the structure initialization and the gradient descent algorithm for the parameter initialization; the second phase is the system simplification which includes the method of removing redundant rules and merging similar rules for the structure simplification and the gradient descent algorithm for the parameter finalization. From the point of view of clustering based system identification, the main contributions of the paper are:

• The special clustering requirement for system identification is recognized and specified. That is, coarser clustering in the regions where the identified system is smoother, and finer clustering in the regions where the identified system is highly variable or nonlinear. Such a special requirement highlights the supervised learning and uneven partition nature for the clustering task faced by the system identification and emphasizes the importance of using the output training data to guide and validate the clustering process and outcome. By pointing out the output training data can be used as the supervisory signals to measure the smoothness and variation of a system to be identified, this recognized clustering requirement for system identification provide a new view and thinking to develop the clustering based system identification methods.

• The FRIOC method is proposed for the structure initialization of FNNs. Comparing with the existing clustering based system identification approaches, the new method has the following advantages: Firstly, in comparing with the one stage clustering approach, it fits perfectly to the special uneven clustering requirement for system identification and overcomes the fundamental weakness of one stage even partition clustering methods; secondly, in comparing with the two stage clustering approach, it overcomes the main difficulties in finding the right number of input clusters for each output context or constraint and clustering the highly overlapped input data sets under different output context or constraints, and provide a structure initialization method which is conceptually simpler, computationally more effective, and highly accurate.

• The accurate similarity analysis and computing method is proposed for the structure simplification of FNNs. Comparing with the existing approximate similarity analysis and computing methods, the proposed method produces more accurate and correct similarity degree between fuzzy rules with Gaussian MF. In this way, it solves the long standing open problem how to accurately compute the similarity degree between fuzzy sets and rules with Gaussian MFs, and provides a simple method which for the first time makes it possible to accurately calculate this type of similarity degrees.

By using several benchmark examples in function approximation and dynamic system identification, the applicability and effectiveness of the proposed method is verified. In particular, it is shown by these benchmark examples that the proposed approach produces simple, compact, and highly accurate FNNs, in comparing with the existing methods. In other words, comparing with the existing approaches, the main advantages of the proposed approach are that it can obtain more accurate and more compact (i.e., simpler) FNNS. However, such advantages are not obtained without a price. Comparing with the most clustering based system identification methods, the proposed approach has the additional phase for structure simplification which requires additional computing steps and times. Therefore, the main disadvantages of our approach is that it is more time-consuming and requires more computing steps and algorithms, comparing with the existing methods without
Acknowledgements

The authors would like to thank the Managing Editor, Handling Editor, and reviewers for their very valuable comments and suggestions which have greatly helped us to improve the quality of the paper.

This work was supported by National Science Foundation of China under Grants 61225016 and 61533002, Beijing Science and Technology Project under Grant Z141100001414005 and Z141101004414058, Beijing Nova Program under Grant Z13110400413007, China Postdoctoral Science Foundation under Grant 2014M550017, Beijing Municipal Education Commission Foundation under Grant km201410005001 and KZ201410005002, and Beijing Municipal Natural Science Foundation under Grant 4144067.

References