

# Redundancy Detection and Removal Tool for Transparent Mamdani Systems

Andri Riid, Kalle Saastamoinen, and Ennu Rüstern

**Abstract.** In Mamdani systems, redundancy of fuzzy rule bases that derives from extensive sharing of a limited number of output membership functions among the rules, is often an overlooked property. In current study, means for detection and removal of such kind redundancy have been developed. Our experiments with case studies collected from literature and Mackey-Glass time series prediction models show error-free rule base reduction by 30-60% that partially cures the curse of dimensionality problem characteristic to fuzzy systems.

## 1 Motivation

One very acute problem that is marring the large scale applications of fuzzy logic is the combinatorial explosion of rules (curse of dimensionality). As the number of membership functions (MFs) and/or input variables increases, the upper bound on the count of fuzzy rules grows exponentially:

$$R_{max} = \prod_{i=1}^N S_i, \quad (1)$$

where  $S_i$  is the number of MFs per  $i$ -th input variable ( $i = 1, \dots, N$ ).

---

Andri Riid

Laboratory of Proactive Technologies, Tallinn University of Technology,  
Ehitajate tee 5, 19086, Tallinn, Estonia  
e-mail: andri@dcc.ttu.ee

Kalle Saastamoinen

Department of Military Technology, National Defence University, P.O. Box 7 FI-00861,  
Helsinki, Finland  
e-mail: kalle.saastamoinen@mil.fi

Ennu Rüstern

Department of Computer Control, Tallinn University of Technology, Ehitajate tee 5, 19086,  
Tallinn, Estonia  
e-mail: ennu.rystern@dcc.ttu.ee

In the ideal fuzzy system the number of fuzzy rules  $R = R_{max}$ , meaning that the rule base of the system is fully defined and contains all possible antecedent combinations. Situation  $R > R_{max}$  indicates a failure in fuzzy system design - either redundant or contradictory rules are present, both of which are the signs of sloppy system design. In real life applications, however, the number of rules often remains well below  $R_{max}$  for several reasons.

First of all, commonly there is not enough material (data) or immaterial (knowledge) evidence to cover the input space universally, not only because it would be too time consuming to collect exhaustive evidence in large scale applications but also because of potential inconsistency that certain antecedent combinations may present (an antecedent “IF sun is bright AND rain is heavy” could be one such example).

Moreover, it is common practice that for the sake of compactness, the rules with little relevance are excluded from the model (for all we know they may be based on few noisy samples). The exclusion decision of a given rule may be based on its contribution to approximation properties (using singular value decomposition, orthogonal transforms, etc. [1]) or on how often or to what degree a given rule is contributing to the output (this, for example, can be easily evaluated by computing cumulative rule activation degrees on available data).

On the whole, rule base reduction can be fitted under two categories - error-free reduction or degrading reduction. Error-free reduction searches for existing redundancies in the model. In other words, if error-free reduction is effective, it is actually an indicator that initial system design was not up to the standard.

With degrading simplification, the model is made less complex by removing non-redundant system parameters. Incidentally, this is achieved at the expense of system universality, accuracy etc.

Typically, reduction is carried out on initial complex model. However, with certain design methodologies unnecessary complexity is avoided by model design procedure. A typical example is the application of tree partitioning of the input space (instead of more common grid partitioning) but the most common constructive compactness-friendly approach these days (related primarily to 1st order Takagi-Sugeno systems [2]) is fuzzy clustering. With clustering, the rules are created in product space only in regions where data concentration is high. Interestingly enough, the side effect of that is the redundancy of cluster projections that are used as the prototypes for MFs of the model. The projections that become fuzzy sets may be highly similar to each other, similar to the universal set or reduced to singleton sets, which calls for adequate methods to deal with that [3]. Another feature of product space clustering is that  $R$  is always a lot smaller than  $R_{max}$  (in fact  $R = S_i$  prior to simplification). For this reason and also from interpolational aspect, product space clustering is not very well suited for Mamdani modeling.

In Mamdani systems, a relatively small set of output MFs is typically shared among rules. This creates substantial redundancy potential, which can be exploited for rule base reduction. For a special class of Mamdani systems (transparent Mamdani systems, more closely observed in Sect. 2) this reduction can actually be error-free, i.e. without any performance loss. In Sect. 3, practical redundancy detection and

removal scenarios have been investigated. Sect. 4 considers the typical implementation issues when designing a computer program for reduction of Mamdani systems. The remainder of the paper presents application examples and performance analysis. Note that the current implementation of the reduction tool can be freely downloaded from <http://www.dcc.ttu.ee/andri/rdart>".

## 2 Transparent Mamdani Systems

Generally, fuzzy rules in Mamdani-type fuzzy systems are based on the disjunctive rule format

$$\begin{aligned} &\text{IF } x_1 \text{ is } A_{1r} \text{ AND } x_2 \text{ is } A_{2r} \text{ AND } \dots \text{ AND } x_N \text{ is } A_{Nr} \text{ THEN } y \text{ is } B_r \\ &\text{OR } \dots \end{aligned} \quad (2)$$

where  $A_{ir}$  denote the linguistic labels of the  $i$ -th input variable associated with the  $r$ -th rule ( $i = 1, \dots, N$ ), and  $B_r$  is the linguistic label of the output variable, associated with the  $r$ -th rule.

Each  $A_{ir}$  has its representation in the numerical domain - the membership function  $\mu_{ir}$  (the same applies to  $B_r$  that is represented by  $\gamma_r$ ) and in general case the inference function that computes the fuzzy output  $F(y)$  of the system (2) has the following form

$$F(y) = \bigcup_{r=1}^R \left( \left( \bigcap_{i=1}^N \mu_{ir}(x_i) \right) \cap \gamma_r \right), \quad (3)$$

where  $\bigcup_r^R$  denotes the aggregation operator (corresponds to OR in (2)),  $\cap$  is the implication operator (THEN) and  $\bigcap_i^N$  is the conjunction operator (AND). In order to obtain crisp output, (3) is generally defuzzified with center-of-gravity method

$$y = Y_{cog}(F(y)) = \frac{\int_Y y F(y) dy}{\int_Y F(y) dy}. \quad (4)$$

The results obtained in current paper are valid for a class of Mamdani systems that satisfy the following requirements:

- The inference operators used here are product and sum. With product-sum inference (4) reduces to

$$y = \frac{\sum_{r=1}^R \tau_r c_r s_r}{\sum_{r=1}^R \tau_r s_r}, \quad (5)$$

where  $\tau_r$  is the activation degree of  $r$ -th rule (computed with the conjunction operator (product)) and  $c_r$  and  $s_r$  are the center-of-gravity and area of  $\gamma_r$ , respectively (see [4]).

- The input MFs ( $s = 1, \dots, S_i$ ) are given by the following definition:

$$\mu_i^s(x_i) = \begin{cases} \frac{x_i - a_i^{s-1}}{a_i^s - a_i^{s-1}}, & a_i^{s-1} < x_i < a_i^s \\ \frac{a_i^{s+1} - x_i}{a_i^{s+1} - a_i^s}, & a_i^s < x_i < a_i^{s+1} \\ 0, & a_i^{s+1} \leq x_i \leq a_i^{s+1} \end{cases}, \quad (6)$$

Such definition of input MFs satisfies input transparency condition assumed for correct interpretation of Mamdani rules (see [5] for further details), however, in current paper we are more interested in its other property, namely

$$\sum_{s=1}^{S_i} \mu_i^s = 1. \quad (7)$$

- The number of output MFs is relatively small and they are shared among rules (this is the usual case in Mamdani systems).

### 3 Error-Free Rule Base Reduction Principles

Consider a pair of fuzzy rules that share the same output MF  $B_\xi$

$$\begin{aligned} & \text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_i \text{ is } A_i^s \text{ ..... AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi \\ & \text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_i \text{ is } A_i^{s+1} \text{ ..... AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi \end{aligned} \quad (8)$$

It is possible to replace these two rules by a single one:

$$\begin{aligned} & \text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_i \text{ is } (A_i^s \text{ OR } A_i^{s+1}) \dots \\ & \dots \text{ AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi \end{aligned} \quad (9)$$

This replacement can be validated very easily, as it derives from (5) that numerically, (8) is represented by (10).

$$\mu_i^s \prod_{j=1, j \neq i}^N \mu_j^{s_j} + \mu_i^{s+1} \prod_{j=1, j \neq i}^N \mu_j^{s_j}. \quad (10)$$

Obviously (10), is equivalent to (11)

$$(\mu_i^s + \mu_i^{s+1}) \prod_{j=1, j \neq i}^N \mu_j^{s_j}, \quad (11)$$

which is nothing else than a representation of (9), assuming that the OR operand is implemented through sum.

This line of logic, while hardly practical for the reduction of fuzzy systems (fuzzy logic software does not usually have any support for such constructions as (9) and numerically, (11) is not really an improvement over (10)), however, has three off-springs (or special cases) that can be really useful as evidenced below.

**Lemma 1.** Consider not a pair but a subset of fuzzy rules consisting of  $S_i$  rules that share the same output MF  $B_\xi$  so that

$$\text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_i \text{ is } A_i^s \dots \text{ AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi \quad (12)$$

$s = 1, \dots, S_i$

Apparently, this would be equivalent to a rule

$$\text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_i \text{ is } (A_i^1 \text{ OR } A_i^2 \text{ OR } \dots \text{ OR } A_i^{S_i}) \dots \quad (13)$$

$\dots \text{ AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi$

We proceed by showing that (13) is equivalent to (14).

$$\text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_{i-1} \text{ is } A_{i-1}^{s_{i-1}} \text{ AND } x_{i+1} \text{ is } A_{i+1}^{s_{i+1}} \dots \quad (14)$$

$\dots \text{ AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi$

*Proof.* For proof we need to show that

$$\sum_{s=1}^{S_i} \mu_i^s \prod_{j=1, j \neq i}^N \mu_j^{s_j} = \prod_{j=1, j \neq i}^N \mu_j^{s_j}, \quad (15)$$

which is valid when

$$\sum_{s=1}^{S_i} \mu_i^s = 1, \quad (16)$$

which is ensured by (6) that concludes the proof.

**Example 1.** Consider three rules of a two-input tipping system:

$$\begin{aligned} &\text{IF } \textit{food} \text{ is } \textit{bad} \text{ AND } \textit{service} \text{ is } \textit{bad} \text{ THEN } \textit{tip} \text{ is } \textit{zero} \\ &\text{IF } \textit{food} \text{ is } \textit{OK} \text{ AND } \textit{service} \text{ is } \textit{bad} \text{ THEN } \textit{tip} \text{ is } \textit{zero} \\ &\text{IF } \textit{food} \text{ is } \textit{good} \text{ AND } \textit{service} \text{ is } \textit{bad} \text{ THEN } \textit{tip} \text{ is } \textit{zero} \end{aligned} \quad (17)$$

If there are no more linguistic labels of food quality as Fig. 1 clearly implies, it is indeed the case that if service is good, output of the system (the amount of tip) is independent from food quality that can be expressed by the following single rule

$$\text{IF } \textit{service} \text{ is } \textit{bad} \text{ AND } \textit{food} \text{ is } \textit{whatever} \text{ THEN } \textit{tip} \text{ is } \textit{zero}, \quad (18)$$

where “whatever” (or “don’t care”) describes the situation that food quality may have any value in its domain without a slightest effect to the output and can thus be removed from the rule, resulting in a nice compressed formulation

$$\text{IF } \textit{service} \text{ is } \textit{bad} \text{ THEN } \textit{tip} \text{ is } \textit{zero} \quad (19)$$

**Lemma 2:** If a subset of fuzzy rules consisting of  $S_i - 1$  rules share the same output MF

|                 |             | food quality |           |             |
|-----------------|-------------|--------------|-----------|-------------|
|                 |             | <i>bad</i>   | <i>OK</i> | <i>good</i> |
| service quality | <i>bad</i>  | zero         | zero      | zero        |
|                 | <i>OK</i>   |              |           |             |
|                 | <i>good</i> |              |           |             |

Fig. 1 Redundancy of rules that makes rule compression possible

$$\text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_i \text{ is } A_i^s \text{ ..... AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi \quad (20)$$

$$s = 1, \dots, S_i, s \neq t$$

then this group of rules can be replaced by a following single rule.

$$\text{IF } x_1 \text{ is } A_1^{s_1} \text{ AND } \dots \text{ AND } x_i \text{ is NOT } A_i^t \text{ ..... AND } x_N \text{ is } A_N^{s_N} \text{ THEN } y \text{ is } B_\xi \quad (21)$$

*Proof.* To prove that we need to show that

$$\sum_{s=1, s \neq t}^{S_i} \mu_i^s \prod_{j=1, j \neq i}^N \mu_j^{s_j} = (1 - \mu_i^t) \prod_{j=1, j \neq i}^N \mu_j^{s_j}, \quad (22)$$

where  $1 - \mu_i^t$  represents the negation of  $A_i^t$ .

It is easy to see that  $\sum_{s=1, s \neq t}^{S_i} \mu_i^s = 1 - \mu_i^t$  if MFs of the  $i$ -th input variable add up to one (7), which completes the proof.

For the remainder of the paper we term replacement schemes (14) and (21) as rule compression scenarios A and B, respectively.

**Example 2:** Consider two rules of a hypothetical fuzzy system

$$\begin{aligned} &\text{IF } \textit{food} \text{ is } \textit{good} \text{ AND } \textit{service} \text{ is } \textit{OK} \text{ THEN } \textit{tip} \text{ is } \textit{large} \\ &\text{IF } \textit{food} \text{ is } \textit{good} \text{ AND } \textit{service} \text{ is } \textit{good} \text{ THEN } \textit{tip} \text{ is } \textit{large} \end{aligned} \quad (23)$$

(23) implies that the amount of tip is independent from service quality if food quality is good and service quality is “anything else than bad” or simply “NOT bad”:

$$\text{IF } \textit{food} \text{ is } \textit{good} \text{ AND } \textit{service} \text{ is NOT } \textit{bad} \text{ THEN } \textit{tip} \text{ is } \textit{large} \quad (24)$$

|                 |             | food quality |           |             |
|-----------------|-------------|--------------|-----------|-------------|
|                 |             | <i>bad</i>   | <i>OK</i> | <i>good</i> |
| service quality | <i>bad</i>  |              |           |             |
|                 | <i>OK</i>   |              |           | large       |
|                 | <i>good</i> |              |           | large       |

Fig. 2 Rule base configuration allowing NOT construction

Note that it would be also possible to write (24) as

$$\text{IF } food \text{ is } good \text{ THEN } tip \text{ is } large \text{ UNLESS } food \text{ is } bad \quad (25)$$

**Lemma 3.** Consider a pair of rules (8) and assume that there are  $R' = \prod_{j=1, j \neq i}^N S_j$  similar pairs that share the output MFs  $B_\xi$  within the pair ( $\xi \in [1, \dots, T]$ ). If this is the case, the MFs  $\mu_i^s$  and  $\mu_i^{s+1}$  can be merged into  $\mu_i^{s \cup s+1} = \mu_i^s + \mu_i^{s+1}$  by the means of summation, consequently each rule pair (8) will reduce to

$$\text{IF } x_1 \text{ is } A_1^{s1} \text{ AND } \dots \text{ AND } x_i \text{ is } A_i^{s \cup s+1} \dots \text{ AND } x_N \text{ is } A_N^{sN} \text{ THEN } y \text{ is } B_\xi \quad (26)$$

*Proof.* Any rule pair (8) is represented by (27) with ( $\xi \in [1, \dots, T]$ ).

$$(\mu_i^s + \mu_i^{s+1}) \prod_{j=1, j \neq i}^N \mu_j^{s_j} \sum_{\xi=1}^{R'} c_\xi s_\xi \quad (27)$$

Obviously, the common term  $\mu_i^s + \mu_i^{s+1}$  can be permanently replaced by  $\mu_i^{s \cup s+1} = \mu_i^s + \mu_i^{s+1}$ .

**Example 3.** The six rules depicted in Fig. 3 can be replaced by three rules

$$\begin{aligned} &\text{IF } food \text{ is } bad \text{ AND } service \text{ is } bad \text{ THEN } tip \text{ is } zero \\ &\text{IF } food \text{ is } bad \text{ AND } service \text{ is } OK \text{ THEN } tip \text{ is } normal \\ &\text{IF } food \text{ is } bad \text{ AND } service \text{ is } good \text{ THEN } tip \text{ is } large \end{aligned} \quad (28)$$

where “bad” is the name for the MF that combines former “very bad” and “bad”.

Note that the merge of two triangles of (6) by sum would result in a trapezoid MF and the updated partition would still satisfy (7).

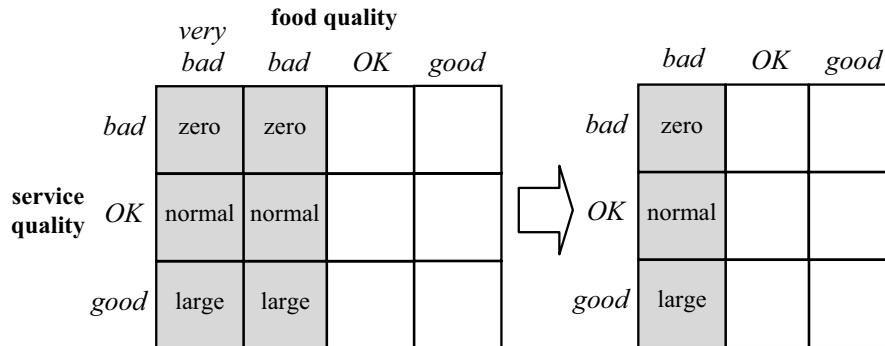


Fig. 3 Redundancy of MFs revealed by rule base analysis

## 4 Implementation

For fuzzy logic software tools the rule base information is generally stored in a separate  $R \times (N + 1)$  dimensional matrix (MATLAB Fuzzy Logic Toolbox uses a variant of this) that accommodates the identifiers of MFs associated with fuzzy rules. Each row in the matrix represents an individual rule and column specifies the input variable (output variable in the last column, which is written in bold in the examples below) to which the identifier in current column is assigned to. Note that NOT-operator is conveniently represented by a minus sign and 0 represents a removed antecedent variable, e.g.  $r$ -th line 1 2 0 -1 **4** would be equivalent to a rule

$$\text{IF } x_1 \text{ is } A_{1r} \text{ AND } x_2 \text{ is } A_{2r} \text{ AND } x_4 \text{ is NOT } A_{1r} \text{ THEN } y \text{ is } B_4 \quad (29)$$

Implementation of rule base reduction schemes described in Sect. 3 is based on the analysis of the rule matrix and subsequent manipulation of it, which is described with the following algorithm (except the detection of redundant MFs that follows directly the logic under Lemma 3).

1. Fix  $i$ -th input variable (e.g. input No. 3 in Fig. 4)
2. Delete (temporarily) the indices corresponding to  $i$ -th variable from the rule matrix.
3. Split the rule matrix into submatrices/subsets of rules so that remaining input variables have fixed MFs throughout the subset.
4. Pick a submatrix.
  - a. If the output MF associated with the rules is the same throughout the submatrix (like in (12)) apply rule compression scenario A by picking one of the rules, inserting zero into the blank spot and deleting the remaining rules.
  - b. If there are two output MFs associated with the rules and one of them is used only once apply rule compression scenario B by picking the rule with the output MF used only once and restoring its deleted index. Then pick one rule

from the rest of the rules and insert negative value of the index just restored to the blank spot and delete the remaining rules.

- c. If none of the above is true just restore the deleted indices within the submatrix.
- 5. Pick another submatrix (step 4) or, alternatively, if all submatrices have been treated, combine submatrices into one rule matrix.
- 6. Pick another variable (step 1) or, alternatively, if all input variables have been treated end.

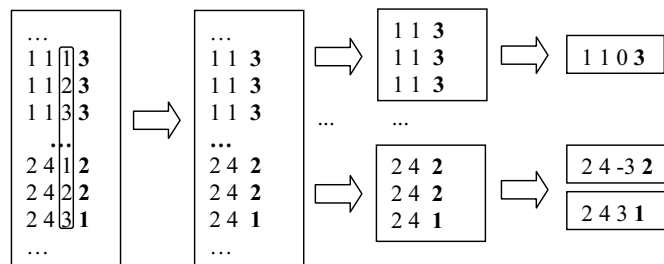


Fig. 4 Principal steps of rule compression algorithm

### 4.1 Higher-Order Compression and Decompression

Though higher-order redundancies are less frequently encountered, the algorithm must be able to handle such situations, i.e. to be able to detect redundancies between already compressed rules. Consider a generous tipping example depicted in Fig. 5.

|                 |             | food quality |           |             |
|-----------------|-------------|--------------|-----------|-------------|
|                 |             | <i>bad</i>   | <i>OK</i> | <i>good</i> |
| service quality | <i>bad</i>  | zero         | zero      | zero        |
|                 | <i>OK</i>   | normal       | large     | large       |
|                 | <i>good</i> | normal       | large     | large       |

Fig. 5 Rule base with higher order redundancies

In the first run we will come up with following compressed rules

$$\begin{aligned}
 & \text{IF } \textit{service} \text{ is bad THEN } \textit{tip} \text{ is zero} \\
 & \text{IF } \textit{food} \text{ is bad AND } \textit{service} \text{ is NOT bad THEN } \textit{tip} \text{ is normal} \\
 & \text{IF } \textit{food} \text{ is OK AND } \textit{service} \text{ is NOT bad THEN } \textit{tip} \text{ is large} \\
 & \text{IF } \textit{food} \text{ is good AND } \textit{service} \text{ is NOT bad THEN } \textit{tip} \text{ is large}
 \end{aligned} \tag{30}$$

For seeking for further compression we run the algorithm once again (or  $N$  times in general case) on compressed rule set. Fig. 6 depicts the rule matrix before (left) and after second compression (right). The rules corresponding to the latter are given by (31).

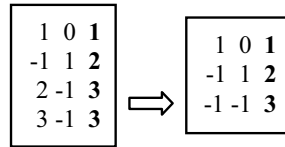


Fig. 6 Rule base reduction with higher order redundancies

$$\begin{aligned}
 & \text{IF } \textit{service} \text{ is bad THEN } \textit{tip} \text{ is zero} \\
 & \text{IF } \textit{food} \text{ is bad AND } \textit{service} \text{ is NOT bad THEN } \textit{tip} \text{ is normal} \\
 & \text{IF } \textit{food} \text{ is NOT bad AND } \textit{service} \text{ is NOT bad THEN } \textit{tip} \text{ is large}
 \end{aligned} \tag{31}$$

The inverse procedure to rule compression - decompression - is even easier to implement. The premise part of the rule base is scanned for zero and negative indices and if such is found, each rule containing a zero index in  $i$ -th position is replaced by  $S_i$  rules so that all indices from 1 to  $S_i$  are represented in the  $i$ -th position. If a negative index is found then  $S_i - 1$  rules are generated, the index at  $i$ -th position running from 1 to  $S_i$ , except for the index that is the absolute value of found negative index. The scan is carried on until there are no more nonzero or negative indices in the rule base. For example, rule (29) has been decompressed in Fig. 7 (we assume that  $S_3 = 3, S_4 = 4$ ). We can see that a deceptively innocent rule can have a large “family” of offsprings when decompressed.

## 4.2 Preserving the Integrity of the Rule Base

Naturally enough, one would expect that the decompression of a compressed rule set would return us to the initial rule base. To ensure that, rule subsets subject to compression must be non-overlapping, i.e. each original rule can only serve once as “raw material” for the compression procedure and cannot be “recycled”. Consider the following, “conservative tipping” example. From the initial rule set in Fig. 8 we can extract a compressed rule “IF *service* is bad THEN *tip* is zero” or “IF *food* is bad THEN *tip* is zero” but not both simultaneously because this means that the rule

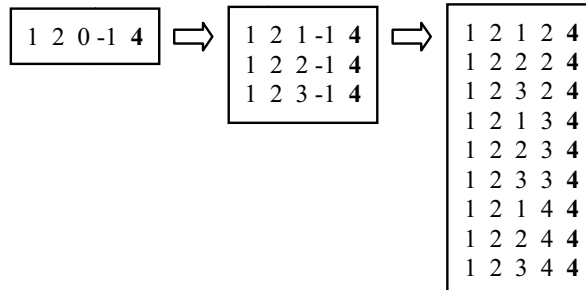


Fig. 7 Inverse procedure to the compression of rules

|                 |             | food quality |           |             |
|-----------------|-------------|--------------|-----------|-------------|
|                 |             | <i>bad</i>   | <i>OK</i> | <i>good</i> |
| service quality | <i>bad</i>  | zero         | zero      | zero        |
|                 | <i>OK</i>   | zero         | normal    | normal      |
|                 | <i>good</i> | zero         | normal    | large       |

Fig. 8 Conflicting simplification scenarios

“IF *service* is *bad* AND *food* is *bad* THEN *tip* is *zero*” has contributed twice for the improved rule base and would be also decompressed twice.<sup>1</sup>

Similarly, if we decide in favor of the first compressed rule then next we choose “IF *service* is *good* AND *food* is *good* THEN *tip* is *large*” and from the remainder we can extract “IF *service* is NOT *bad* and *food* is *OK* THEN *tip* is *normal*” but not simultaneously with “IF *service* is *OK* AND *food* is NOT *bad* THEN *tip* is *normal*”. Due to the need to validate the compressions, the execution of rule compression becomes much more complicated than could be understood in first place and is controlled with the following mechanism described below.

When we fix an input variable, and extract a subset of rules, feasibility of compression is verified first against the internal conditions within the subset. When this test is passed, necessary compression is carried out and its feasibility is verified against the initial rule base (by simply looking for duplicates between the initial rule base and decompressed compressed subset). If the duplicates are found, the

<sup>1</sup> These two rules together are technically equivalent to “IF *service* is *bad* OR *food* is *bad* THEN *tip* is *zero*”, Our reasoning above implies that disjunctive antecedents are prohibited.

subset is returned to the initial set and next subset is extracted. Only if there are no duplicates, the compression is actually executed, the compressed rule is added to the set of compressed rules (if available). The source rules, however, are returned to the working rule set. After that a new subset is handled. In the end of the cycle (all input variables have been picked one by one) the working rule set minus all these rules that were used for the compression, becomes a so-called reserve set (which will be temporarily added to the working rule set when duplicates are being sought) and the set of compressed rules becomes a new working rule set for higher order compression. In the very end (the rule compression algorithm has been applied  $N$  times) the reserve set plus the compressed set of  $N$ -th cycle become the final rule base.

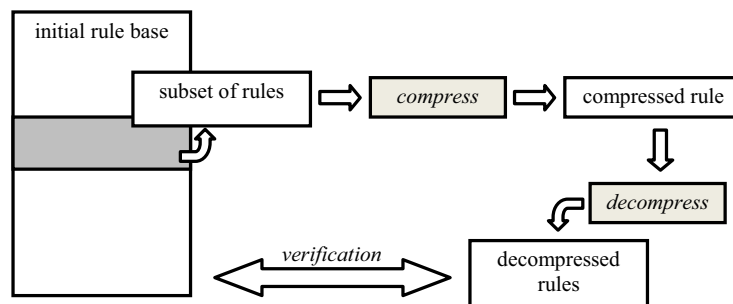


Fig. 9 Compression validation

### 4.3 Incomplete Rule Bases

The reasoning throughout the section so far is based on the assumption that the rule base is complete. Yet in many applications we must deal with incomplete rule bases (the number of rules  $R < \prod_{i=1}^N S_i$ ). Incompleteness of the rule base arises the question how to treat the blank spots in the rule base. Could we use them to our advantage so as to minimize the number of rules? Or should we maintain status quo and integrity of the rule base? To explain this dilemma in finer detail let us look at another tipping system in Fig. 10 that has two undefined rules. In first (optimistic) approach where we are about to make advantage of rule base incompleteness, we would end up with five rules including two compressed rules "IF *service* is *bad* THEN *tip* is *zero*" and "IF *service* is *OK* THEN *tip* is *normal*", ignoring the fact that it would actually mean writing *zero* and *normal*, respectively, into two blank spots and thus changing the original rule base. In conservative approach, though the number of rules would be the same after compression, two compressed rules would be "IF *service* is *bad* AND *food* is NOT *OK* THEN *tip* is *zero*" and "IF *service* is *OK* AND *food* is NOT *bad* THEN *tip* is *normal*", leaving the blank

spots unmodified. Conservative approach seems somewhat closer to the spirit of error-free reduction, however, when we look at the problem at numerical level, the pros and cons are not so obvious.

|                 |             | food quality |           |             |
|-----------------|-------------|--------------|-----------|-------------|
|                 |             | <i>bad</i>   | <i>OK</i> | <i>good</i> |
| service quality | <i>bad</i>  | zero         |           | zero        |
|                 | <i>OK</i>   |              | normal    | normal      |
|                 | <i>good</i> | zero         | normal    | large       |

**Fig. 10** An incomplete rule base: another challenge

Each undefined rule means that there is some area in the input space for which we cannot compute matching output values. In practice some pre-specified value (usually average of the domain of the output variable) is used in this case to maintain continuity. If we use the blank spot to our advantage we typically use a neighboring rule as the prototype. Both may and may not be adequate guesses for the missing rule, neither is clearly better. Therefore, the simplification tool must have a built-in option to determine if we take optimistic or conservative approach when treating incomplete rule bases. In the following, we use the notation  $I_c = 1$  for the conservative and  $I_c = 0$  for the optimistic approach.

Taking above considerations into account, rule compression algorithm becomes more complex. Whereas with optimistic strategy nothing changes - we can apply rule compression scenario A when there is one output MF throughout the subset (see the algorithm in Sect. 4) and scenario B when there are two (and one of them is used only once), with conservative strategy we must also take into account how many rules are there in the subset. The selection map given in Table 1, where  $N_o$  denotes the number of unique output MFs within the subset (once again, 2 MFs must satisfy the condition than one of them cannot be used more than once).

**Table 1** Selecting between different compression scenarios (conservative strategy)

| $N_o$ | $S_i$ rules | $S_i - 1$ rules |
|-------|-------------|-----------------|
| 1     | A           | B               |
| 2     | B           | -               |

## 5 Applications

The proposed approach is validated on three applications from literature that come from different areas of engineering - truck backer-upper control, skin permeability modeling and fuzzy decision support systems. Additionally, a thorough experiment on simplifying Mackey-Glass time series prediction models is carried out to provide analysis material.

### 5.1 Simplification of Systems from Literature

In first case study the simplification algorithm was applied to the fuzzy trajectory management unit (TMU) of truck backer-upper control system from [6] that originally uses 28 rules that specify the optimal truck angle  $\Phi_r$  in respect to its coordinates  $x$  and  $y$ . Application of the algorithm reveals that the original controller is heavily redundant as the number of its rules can be reduced to 11 without any loss in control quality that means almost 60% reduction in size (see Fig. 11). Incidentally, the biggest contribution to size reduction comes from detection and merging redundant MFs (13 rules), rule compression scenario A removes 2 and scenario B further 2 rules. As the original rule base is complete (as it is the case with two remaining case studies), applying the simplification tool with either (optimistic or conservative) option produces the same exact result.

In second case study, we undertook the task of reducing the rule base of the model developed in [10] that has octanol-water partition coefficient ( $\log K_{ow}$ ), molecular weight ( $M_w$ ) and temperature ( $T$ ) as its inputs and skin permeability coefficient ( $\log K_p$ ) as the output. The inputs are partitioned into 4, 3 and 3 fuzzy sets, respectively and output has three MFs. Because the MFs in [10] do not satisfy (7) (custom MFs functions were used in the application, moreover, for this system a special inference scheme has been developed because the model is expected to behave as a classifier) rule base reduction would not be error-free but nevertheless, the number of rules could potentially be reduced to 20 from 36 (56% reduction) - 8 by rule compression scenarios A and B each.

In third case study, the simplification tool was applied to the fuzzy decision support system [11] that has three inputs - detection, frequency and severity (all of which have been partitioned into five fuzzy sets), an output - fuzzy risk priority category - that has nine MFs and 125 rules. We are able to bring the rule count down to 75 (40% reduction) - out of which 25 disappear by merging two of the MFs of severity, 16 can be compressed by scenario A and further 9 by scenario B.

### 5.2 Mackey-Glass Time Series Prediction

The example includes prediction of time series that is generated by the Mackey-Glass [7] time-delay differential equation

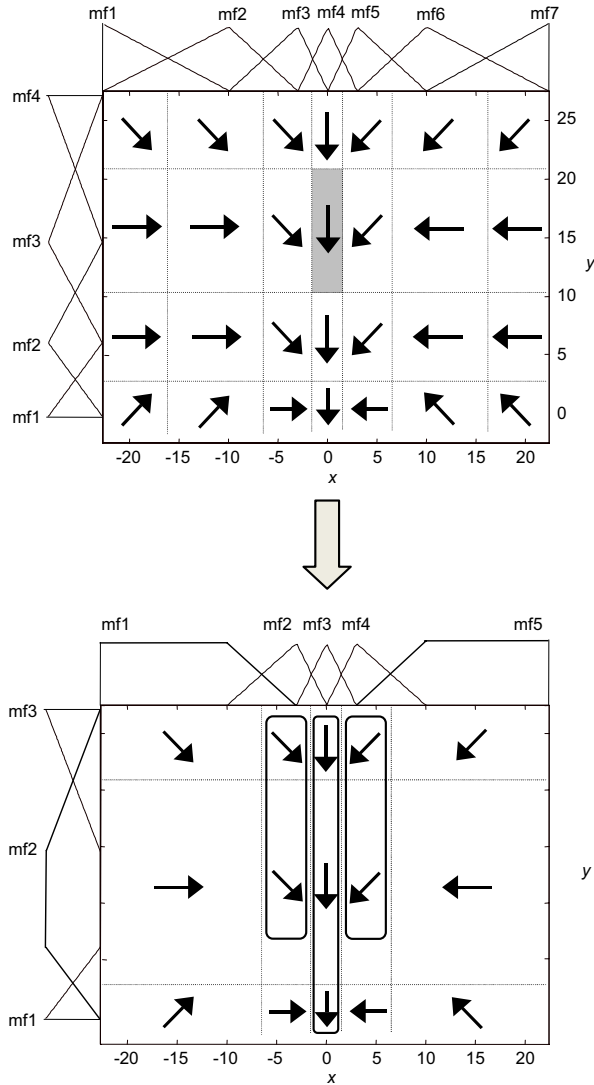


Fig. 11 TMU of the truck backer-upper before (above) and after (below) the simplification

$$\dot{x} = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)} - 0.1x(t), \tag{32}$$

and subsequent simplification of the prediction model.

To obtain the time series value at integer points, the numerical solution to the above MG equation is found using the fourth-order Runge-Kutta method. We assume  $x(0) = 1.2$ ,  $\tau = 17$ , and  $x(t) = 0$  for  $t < 0$ .

We use up to 4 known values of the time series in time, to predict the value in the future. For each  $t$ , the input training data is a four-dimensional vector of the following form.

$$x(t+6) = f(x(t-18), x(t-12), x(t-6), x(t)) \quad (33)$$

There are 1000 input/output data values. We use the first 500 data values for training, while the others are used as checking data for validating the identified fuzzy model.

To obtain the prediction model, we apply a simplistic modeling algorithm [8] that provides a crude predictor of the phenomenon (we are more interested in the performance of our simplification algorithm than in modeling accuracy). This method assumes predefined input-output partition - we are using an uniform one for the sake of simplicity - and finds out the best matching set of fuzzy rules for this partition on the basis of training data samples. For each potential rule we identify the sample  $[x_1(k)x_2(k)x_3(k)x_4(k)y(k)]$  that yields maximum rule activation degree  $\tau_r(k)$  and use  $y(k)$  to determine matching output MF  $\gamma_j, j \in 1, \dots, T$  (the one that produces  $\max(\gamma_j(y(k)))$ ).

To observe the effects of simplification we employ models of different sizes by varying the number of MFs (and even input variables - both 3- and 4-input models are being used). The results are given in Table 2, where first column specifies the  $S_i$ s of each input variable, the second the number of output MFs ( $T$ ). Further columns contain modeling errors on training and checking data ( $\varepsilon_{tr}$  and  $\varepsilon_{ch}$ , respectively), the number of rules before ( $R_0$ ) and after simplification ( $R_f$ ) and rule reduction rates ( $\eta$ ).

The results reveal some general characteristics of the simplification algorithm. It can be seen that rule reduction rate is higher if the number of output MFs is small and number of input MFs is high, which is rather logical, because with a small number of output MFs these are shared more extensively and thus redundancy is more likely to exist. Large number of input MFs on the other hand means that the number of rules that can be replaced by a single rule is generally larger. However, with incomplete rule bases, the large number of input MFs increases the number of undefined rules thus limiting algorithm's capability when we take conservative approach regarding completeness of the rule base. It is, however, clearly evident that this option does not have any effect to the modeling error neither with training nor checking data.

It also turns out that redundancy of input MFs is a rather rare phenomenon as it was detected in none of the above models. Rule compression scenario A contributes more to  $\eta$  if  $I_c = 1$  and scenario B is mostly responsible for redundancy removal if  $I_c = 0$ .

As for comparison of the modeling accuracy - ANFIS [9] with two generalized bell membership functions on each of the four inputs and containing 16 Takagi-Sugeno rules shows  $\varepsilon_{tr} = \varepsilon_{ch} = 0.0025$  after 10 training epochs.

**Table 2** results of MG time series prediction and model simplification

| $S_i$                          | $T$ | $\epsilon_{tr}$ | $\epsilon_{ch}$ | $R_0$ | $R_f$ | $I_c$ | $\eta$ |
|--------------------------------|-----|-----------------|-----------------|-------|-------|-------|--------|
| $4 \times 4 \times 4$          | 5   | 0.0691          | 0.0687          | 57    | 29    | 1     | 49.1%  |
| $4 \times 4 \times 4$          | 5   | 0.0691          | 0.0687          | 57    | 26    | 0     | 54.4%  |
| $4 \times 4 \times 4$          | 9   | 0.0623          | 0.0620          | 57    | 44    | 1     | 22.8%  |
| $4 \times 4 \times 4$          | 9   | 0.0623          | 0.0620          | 57    | 40    | 0     | 30.0%  |
| $5 \times 5 \times 5$          | 5   | 0.0599          | 0.0593          | 88    | 56    | 1     | 36.4%  |
| $5 \times 5 \times 5$          | 5   | 0.0599          | 0.0593          | 88    | 37    | 0     | 58.0%  |
| $5 \times 5 \times 5$          | 9   | 0.0426          | 0.0420          | 88    | 79    | 1     | 10.2%  |
| $5 \times 5 \times 5$          | 9   | 0.0426          | 0.0420          | 88    | 60    | 0     | 31.8%  |
| $6 \times 6 \times 6$          | 5   | 0.0483          | 0.0479          | 128   | 112   | 1     | 12.5%  |
| $6 \times 6 \times 6$          | 5   | 0.0483          | 0.0479          | 128   | 60    | 0     | 53.1%  |
| $6 \times 6 \times 6$          | 9   | 0.0347          | 0.0346          | 128   | 128   | 1     | 0%     |
| $6 \times 6 \times 6$          | 9   | 0.0347          | 0.0346          | 128   | 90    | 0     | 29.7%  |
| $3 \times 3 \times 3 \times 3$ | 5   | 0.0639          | 0.0627          | 68    | 36    | 1     | 47.1%  |
| $3 \times 3 \times 3 \times 3$ | 5   | 0.0639          | 0.0627          | 68    | 41    | 0     | 39.7%  |
| $3 \times 3 \times 3 \times 3$ | 9   | 0.0619          | 0.0607          | 68    | 43    | 1     | 36.8%  |
| $3 \times 3 \times 3 \times 3$ | 9   | 0.0619          | 0.0607          | 68    | 44    | 0     | 35.3%  |
| $4 \times 4 \times 4 \times 4$ | 5   | 0.0384          | 0.0376          | 181   | 110   | 1     | 39.2%  |
| $4 \times 4 \times 4 \times 4$ | 5   | 0.0384          | 0.0376          | 181   | 96    | 0     | 47.0%  |
| $4 \times 4 \times 4 \times 4$ | 9   | 0.0404          | 0.0397          | 181   | 131   | 1     | 27.6%  |
| $4 \times 4 \times 4 \times 4$ | 9   | 0.0404          | 0.0397          | 181   | 115   | 0     | 36.5%  |
| $5 \times 5 \times 5 \times 5$ | 5   | 0.0451          | 0.0442          | 275   | 227   | 1     | 17.5%  |
| $5 \times 5 \times 5 \times 5$ | 5   | 0.0451          | 0.0442          | 275   | 128   | 0     | 53.5%  |
| $5 \times 5 \times 5 \times 5$ | 9   | 0.0354          | 0.0345          | 275   | 254   | 1     | 7.6%   |
| $5 \times 5 \times 5 \times 5$ | 9   | 0.0354          | 0.0345          | 275   | 162   | 0     | 41.1%  |

## 6 Conclusions

In this paper we presented the basis and working principles for the tool for redundancy detection and removal for a special class of Mamdani systems and also demonstrated that the implementation of these ideas indeed reduces complexity of fuzzy systems from different areas of engineering by 30-60% without any loss of accuracy. The major factors that influence the reduction rate by rule compression are the number of input MFs (positive correlation) and the number of output MFs (negative correlation) In certain cases (if the number of input variables is relatively small) MF redundancy may also be the case. Additionally, it was found out that with optimistic strategy, scenario A is mostly responsible for rule base reduction, whereas scenario B plays the key role with conservative strategy; the latter also tends to be ineffective if the number of input variables is large.

However, there may be cases such as the one depicted in Fig. 12, which is a primitive version of standard McVicar-Whelan rule base [12] where, even if the number of unique output MFs is well below the number of rules implying output MF sharing and redundancy potential, the homogeneous rule subsets are oriented so that it

|              |          | <b>change in error</b> |          |          |
|--------------|----------|------------------------|----------|----------|
|              |          | <i>N</i>               | <i>Z</i> | <i>P</i> |
| <b>error</b> | <i>N</i> | NB                     | NS       | Z        |
|              | <i>Z</i> | NS                     | Z        | PS       |
|              | <i>P</i> | Z                      | PS       | PB       |

**Fig. 12** Compact version of McVicar-Whelan rule base where N stands for "negative", P for "positive", Z for "zero", PB for "positive big", PS for "positive small", NS for "negative small" and NB for "negative big".

is really impossible to apply any scheme of rule compression or MF merge. Consequently, orthogonal orientation of homogeneous rule subsets is another, somewhat hidden but nevertheless important requirement for redundancy removal. Investigation, if there are means for redundancy reduction for such fuzzy rule bases is a matter of future research.

**Acknowledgements.** This work has been partially supported by Estonian Science Foundation, grant no. 6837.

## References

1. Yen, J., Wang, L.: Simplifying Fuzzy Rule-Based Models Using Orthogonal Transformation Methods. *IEEE Trans. Systems, Man, Cybern. Part B* 29(1), 13–24 (1999)
2. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Systems Man and Cybern.* 15, 116–132 (1985)
3. Roubos, H., Setnes, M.: Compact and Transparent Fuzzy Models and Classifiers Through Iterative Complexity Reduction. *IEEE Trans. Fuzzy Systems* 9(4), 516–524 (2001)
4. Riid, A., Rüstern, E.: On the Interpretability and Representation of Linguistic Fuzzy Systems. In: *Proc. IASTED International Conference on Artificial Intelligence and Applications*, Benalmadena, Spain, pp. 88–93 (2003)
5. Riid, A., Rüstern, E.: Transparent Fuzzy Systems in Modeling and Control. In: Casillas, J., Cordon, O., Herrera, F., Magdalena, L. (eds.) *Interpretability Issues in Fuzzy Modeling*, pp. 452–476. Springer, New York (2003)
6. Riid, A., Rüstern, E.: Fuzzy logic in control: truck backer-upper problem revisited. In: *Proc. IEEE Int. Conf. Fuzzy Systems*, Melbourne, Australia, vol. 1, pp. 513–516 (2001)
7. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. *Science* 197, 287–289 (1977)

8. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. *IEEE Trans. on Systems, Man and Cybern.* 22(6), 1414–1427 (1992)
9. Jang, J.-S.R.: ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. on Systems, Man and Cybern.* 23(3), 665–685 (1993)
10. Keshwania, D.R., Jonesb, D.D., Meyerb, G.E., Brand, R.M.: Rule-based Mamdani-type fuzzy modeling of skin permeability. *Applied Soft Computing* 8(1), 285–294 (2008)
11. Puente, J., Pino, R., Priore, P., Fuente, D.D.L.: A decision support system for applying failure mode and effects analysis. *Int. J. Quality and Reliability Mgt* 19(2), 137–150 (2002)
12. MacVicar-Whelan, P.J.: Fuzzy Sets for Man-Machine Interaction. *Int. J. Man-Machine Studies* 8, 687–697 (1976)

