A Graph Theory Approach to Fuzzy Rule Base Simplification

Caro Fuchs[1(✉)], Simone Spolaor[2,3], Marco S. Nobile[1,3], and Uzay Kaymak[1]

[1] School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
{c.e.m.fuchs,m.s.nobile,u.kaymak}@tue.nl

[2] Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milan, Italy
simone.spolaor@disco.unimib.it

[3] SYSBIO.IT Centre for Systems Biology, Milan, Italy

Abstract. Fuzzy inference systems (FIS) gained popularity and found application in several fields of science over the last years, because they are more transparent and interpretable than other common (black-box) machine learning approaches. However, transparency is not automatically achieved when FIS are estimated from data, thus researchers are actively investigating methods to design interpretable FIS. Following this line of research, we propose a new approach for FIS simplification which leverages graph theory to identify and remove similar fuzzy sets from rule bases. We test our methodology on two data sets to show how this approach can be used to simplify the rule base without sacrificing accuracy.

Keywords: Fuzzy logic · Takagi–Sugeno fuzzy model · Data-driven modeling · Open-source software · Python · Graph theory

1 Introduction

Fuzzy Inference Systems (FIS) are based on the fuzzy set theory introduced by Zadeh [37]. FIS are universal approximators that can implement non-linear mappings between inputs and output, designed to model linguistic concepts. Owing to these characteristics, FIS have been successfully applied in a variety of fields, including systems biology, automatic control, data classification, decision analysis, expert systems, and computer vision [7,14,21,25,27,30]. One of the main advantages provided by FIS over black-box methods, such as (deep) neural networks, is that fuzzy models are (to a certain degree) transparent, and hence open to interpretation and analysis.

The first FIS relied on the ability of fuzzy logic to model natural language and were developed using expert knowledge [23]. The knowledge of human experts was extracted and transformed into rules and membership functions. These FIS are easy to interpret, but unfortunately, they cannot be easily used to model large and complex systems, since human knowledge is often incomplete and episodic.

In 1985, Takagi and Sugeno [33] proposed a method to construct self-learning FIS from data. The fuzzy rules underlying this kind of FIS are automatically generated from data, but follow the same if–else structure as the rules based on expert knowledge, thus making it possible to model large and complex systems. However, there is generally a loss of semantics when the FIS is constructed in this way, since the number of induced rules can be large, and the rules might become complex because of the number of considered variables. Therefore, many researchers are investigating the problem of designing interpretable fuzzy models [1,2,12].

When FIS are identified from data, it is common to obtain a system with a large number of highly overlapping fuzzy sets, that hardly allow for any interpretation. This hinders the user from labeling the fuzzy sets with linguistic terms and thus giving semantic interpretation to the model. This problem arises especially when Takagi and Sugeno (TS) [33] fuzzy models are determined based on input–output product space fuzzy clustering. Fuzzy rule base simplification has been proposed to reduce the complexity of such models in order to make them more amenable to interpretation [29].

In this paper we propose a new approach based on graph theory to simplify the fuzzy rule base by reducing the number of fuzzy sets in the model when a high overlap is detected between membership functions. Specifically, we combine Jaccard similarity and graph theory to determine which fuzzy sets can be simplified in the model. We name our approach Graph-Based Simplification (GRABS).

GRABS was implemented using the Python programming language [26], and it is part of pyFUME, a novel Python package developed to define FIS from data [9]. pyFUME provides a set of classes and methods to estimate the antecedent sets and the consequent parameters of TS fuzzy models. This information is then used to create an executable fuzzy model using the Simpful library [31], a Python library designed to handle fuzzy sets, fuzzy rules and perform fuzzy inference. pyFUME's source code and documentation can be downloaded from GITHUB at the following address: https://github.com/CaroFuchs/pyFUME.

In this study we investigate the pyFUME's GRABS functionality, testing the methodology on both synthetic and real data sets. Our results show that pyFUME produces interpretable models, written in a human-readable form, characterized by a tunable level of complexity in terms of separation of fuzzy sets.

The paper is structured as follows. We provide a theoretical background about FIS simplification in Sect. 2. The GRABS method is described in Sect. 3. Section 4 describes how to use GRABS in pyFUME. Some results of GRABS with pyFUME are shown in Sect. 5. We conclude the paper in Sect. 6.

2 Rule Base Simplification

When dealing with the interpretability of a fuzzy model, two main aspects have to be considered [1,2]: the readability and comprehensibility. The former depends on the complexity of the FIS structure, while the latter is tied to the semantics associated to it.

Rule base simplification is an approach to simplify the structure of a fuzzy system. Following the classification proposed in [15], methods for fuzzy rule base simplification can be divided into five categories:

1. **Feature reduction.** This category includes methods that rely on feature reduction by means of feature transformation [20] (also referred to as feature extraction) or feature selection [13]. Feature transformation consists of creating additional features from the given ones, or selecting a new set of features to replace the old one. Since feature transformation changes the underlying meaning of the features, this approach can make feature interpretation harder, ultimately resulting in a loss of semantics. Feature selection is not affected by this shortcoming, since it selects a subset of the most influential features, and discards features affected by noise or that do not contribute significantly to the accuracy of the FIS.
2. **Similarity-based simplification.** Methods belonging to this class perform a merging of similar rules and/or eliminate redundancy in the FIS. Similarity merging methods perform a merging of fuzzy sets representing comparable and analogous concepts, by exploiting some similarity measure (see e.g., [6,17,29]). When the model shows high redundancy, this merging might result in some rules being equivalent and thus amenable to being merged, thereby reducing the number of rules as well. Compatible cluster merging algorithms try to combine similar clusters into a single one, in order to reduce the FIS rule base (e.g., [18]). Finally, methods for consistency checking [34] and inactivity checking [17] are employed to decrease the number of rules. In particular, consistency checking reduces the rule base by eliminating conflicting rules, while inactivity checking removes rules with a low firing strength, according to a predetermined threshold.
3. **Orthogonal transformation.** These methods reduce fuzzy rule bases by means of matrix computations. They achieve such reduction in two ways: either by taking into account the firing strength matrix and employing some metrics to estimate the impact of a rule on the FIS performance [36]; or by considering matrix decompositions (e.g., singular value decomposition) and removing the rules that correspond to the less important, smaller components and updating the membership functions accordingly [35].
4. **Interpolative reasoning.** Traditional fuzzy reasoning methods require the universe of discourse of input variables to be entirely covered by the fuzzy rule base. These methods do not perform well when input data fall in a non-covered region of the universe of discourse, as this does not trigger the firing of any rule and no consequences are drawn from the rule base. The first fuzzy interpolative reasoning method was proposed in [19] to overcome the above

mentioned limitation. This method consists in generating the conclusions of FIS with sparse rule bases by means of approximation. In the context of FIS simplification, fuzzy interpolation can be employed to reduce fuzzy rule bases. This is achieved by eliminating the rules that can be approximated through interpolation of neighboring rules.

5. **Hierarchical reasoning.** Methods adopting a hierarchical reasoning approach reorganize the fuzzy rule base structure in order to obtain a hierarchical fuzzy system [28]. Hierarchical fuzzy systems consist of several low-dimensional FISs, connected together according to some defined hierarchy. This approach was applied for example in [32], where the authors propose a hierarchical fuzzy system for the automatic control of an unmanned helicopter.

The GRABS approach proposed in this paper aims at simplifying the fuzzy rule base by eliminating redundant information from the overlapping fuzzy sets. Thus, our approach falls in the category of similarity-based rule base simplification. Methods belonging to this category need to assess the similarity between the fuzzy sets in the antecedents with a given measure, in order to remove similar sets. In [29], the authors suggest to adopt the Jaccard similarity index [16] to quantify such similarity between two fuzzy sets. Given two fuzzy sets A and B, the Jaccard index S is computed as follows:

$$S(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{1}$$

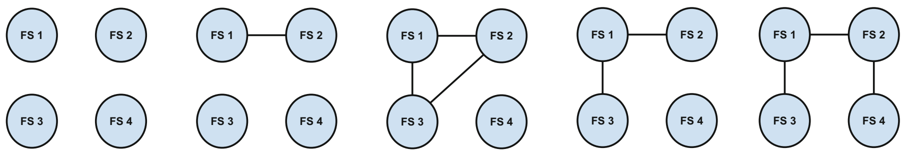
where $|\cdot|$ denotes the cardinality of a fuzzy set, and the $\cap$ and $\cup$ operators represent the intersection and the union of fuzzy sets, respectively. The Jaccard similarity index takes values between 0 and 1, with 1 representing total similarity (i.e, perfectly overlapping fuzzy sets) and 0 disjoint fuzzy sets.

3 Graph-Based Rule Base Simplification

A graph is an abstract mathematical structure used to model pairwise relations between objects. A undirected graph is defined by a pair $G = (V, E)$, where V is a set of vertices (or nodes) connected by a set of edges E.

We represent the similarities between fuzzy sets of a same variable by using a graph. Specifically, each vertex $v \in V$ represents a fuzzy set. See for example Fig. 1a, where four fuzzy sets are defined on the universe of discourse. If the Jaccard similarity of two fuzzy sets exceeds a certain, user-specified, threshold σ, their two nodes are connected by an edge. This process is schematized in Fig. 1b, where the fuzzy sets 1 and 2 show high similarity and therefore are connected by an edge. Please observe that the graph now contains multiple connected components (three in this example).

Assume now that the fuzzy set 3 is also similar to the fuzzy sets 1 and 2. Then, by adding the corresponding additional edges $(3, 1)$ and $(3, 2)$, the graph changes as schematized in Fig. 1c. In particular, the largest component of the graph is



(a) No fuzzy sets show high similarity.

(b) Fuzzy set 1 and 2 show high similarity.

(c) Fuzzy set 1, 2 and 3 all show high similarity to one another.

(d) Fuzzy set 1 and 2 and fuzzy set 1 and 3 show high similarity, but fuzzy set 2 and 3 are dissimilar.

(e) Fuzzy set 1 and 2, 2 and 3, and 3 and 4 show high similarity, but the other sets are dissimilar.

Fig. 1. Graphs representing variables that each have four fuzzy sets.

complete, meaning that clusters 1, 2 and 3 are all similar to one another with respect to the specified threshold σ. Please note that the transitive closure is in general not valid in this context. For example, Fig. 1d shows a case where fuzzy set 1 is similar to both fuzzy set 2 and 3, but fuzzy set 2 and 3 are not similar to each other, according to Jaccard similarity. In this respect, the GRABS method deviates from the compatible cluster merging in [18], where transitive closure is imposed before merging, or from the similarity-based rule base simplification method in [29], where merging takes place iteratively, by combining only the most similar pair of fuzzy sets at each step.

According to our merging algorithm, the fuzzy sets of Fig. 1a should all be retained, since they are all dissimilar. However, in Fig. 1b both fuzzy set 1 and and 2 give the same information. Therefore, one of them can be dropped without losing (much) information and accuracy of the model. The same applies to Fig. 1c: only one of the three similar fuzzy sets can be retained to preserve all the information and the accuracy of the model. In Fig. 1d, dropping fuzzy set 1, 2 or 3 would lead to a loss of information, since fuzzy set 2 and 3 are dissimilar. Therefore, all three fuzzy sets should be retained. We also considered the possibility of multiple partially overlapping fuzzy sets (see Fig. 1e): in this circumstance, we do not allow the full removal of inner nodes characterized by a higher degree (i.e., FS1 and FS2), since that could lead to a fuzzy partitioning that does not span the full universe of discourse. These concepts represents the foundations of our graph-based rule base simplification algorithm. The aforementioned heuristic–that represents a trade-off between simplicity, computational costs, and accuracy in the simplification–seems to be effective for practical scenarios.

The pseudo-code of our GRABS methodology is shown in Listing 1.1. The algorithm begins by creating two empty dictionaries that will store the information about pairs of similar fuzzy sets (line 1) and the information about fuzzy sets replacements (line 2). Then, for each variable, we calculate the pair-wise Jaccard similarities of the associated fuzzy sets (lines 3–13). If the Jaccard similarity of a pair of fuzzy sets is above the user-defined threshold σ then that pair is added to the dictionary (lines 7–10). After the Jaccard similarities are assessed, the algorithm proceeds to build and analyze the graphs. Specifically, the algorithm

Listing 1.1. Pseudocode of pyFUME's GRABS algorithm.

```
1  similar_pairs ← {}
2  replacement ← {}
3  foreach variable in variables:
4      similar_pairs[variable] ← []
5      for fs1 ← 1 to num_fuzzysets:
6          for fs2 ← fs1+1 to num_fuzzysets:
7              similarity ← Jaccard_similarity(fs1, fs2)
8              if similarity > $\sigma$ then:
9                  similar_pairs[variable] ← (fs1, fs2)
10             end if
11         end for
12     end for
13 end foreach
14 foreach variable, similar_clusters in similar_pairs:
15     G ← create_graph(similar_pairs)
16     SC ← G.get_components()
17     for component in SC:
18         if component.is_complete() then:
19             retained ← component.pick_one_node()
20             component.remove_node(retained)
21             foreach node in component
22                 replacement[(variable, node)] ← retained
23             end foreach
24         end if
25     end for
26 end foreach
```

iterates on variables (lines 14–26). For each variable, a graph is created by using the fuzzy sets stored in the `similar_pairs` dictionary (line 15). Then, all the components of the the graph are extracted (line 16) and, for each sub-component, the algorithm performs a completeness check (line 18). If the sub-component is complete, then all fuzzy sets are similar and can be simplified: one node is picked to be retained (line 19) and removed from the component (line 20). The remaining nodes (i.e., similar fuzzy sets) can be removed from the model. We store the information about the removed fuzzy sets in the `replacement` dictionary. To simplify the lookup in pyFUME, the keys of the dictionary are pairs (variable, removed fuzzy set) and the values are the retained fuzzy sets (line 22).

4 GRABS in pyFUME

pyFUME was designed to have an easy to use interface both for practitioners and researchers. Currently, pyFUME supports the following features.

1. Loading the input data.
2. Division of the input data into a training and test data set.

3. Clustering of the data in the input-output space by means of Fuzzy C-Means (FCM) clustering [4] or an approach based on Fuzzy Self-Tuning Particle Swarm Optimization (FST-PSO [8,24]).
4. Estimating the antecedent sets of the fuzzy model, using the method described in [10]. Currently, Gaussian (default option), double Gaussian and sigmoidal membership functions are supported
5. Estimating the consequent parameters of the first-order TS fuzzy model, implementing the functionalities described in [3].
6. The generation, using the estimated antecedents and consequents, of an executable fuzzy model based on Simpful, possibly exporting the source code as a separate, executable file.
7. Testing of the estimated fuzzy model, by measuring the Root Mean Squared Error (RMSE), Mean Squared Error (MSE) or Mean Absolute Error (MAE).

To use pyFUME to estimate a fuzzy model from data, the user simply has to call the `pyfume()` function and specify the path to the data and the number of clusters as input. Optionally, the user can diverge from default settings (for example to use a clustering approach based on FST-PSO [24] or normalizing the data) by choosing additional key-value pairs. More information on pyFUME's functionalities can be found in [9].

If a user wants to use GRABS in pyFUME to simplify the produced rule bases, an optional input argument `similarity_threshold` must be specified. Thanks to this parameter, the user can set any arbitrary threshold for fuzzy sets similarity, implicitly controlling the error tolerance. This allows our method to be applied, in principle, to any system. By default, this threshold is set to 1.0, which means that one of the fuzzy sets is only dropped if the Jaccard similarity (which is assessed by using (1)) is 1.0, i.e., the fuzzy sets are identical. Since membership functions estimated from data will hardly ever be identical, this means that by default the functionality is switched off. The user can activate the functionality by setting `similarity_threshold` to a lower number. For example, when `similarity_threshold` = 0.9, pairs of fuzzy sets that have a Jaccard similarity higher than 0.9 will be dropped.

Internally, pyFUME represents the relation between the fuzzy sets as graphs such as the ones shown in Fig. 1. After identifying all complete components, one vertex for each of them is randomly selected to be retained, while the others are dropped.

In practice, this means that from all overlapping fuzzy sets in the FIS, one set is randomly selected and retained in the model. The other overlapping sets are discarded and remapped to the fuzzy set that was retained. This means that multiple rules in a FIS can have the same fuzzy set for a certain variable in their antecedent.

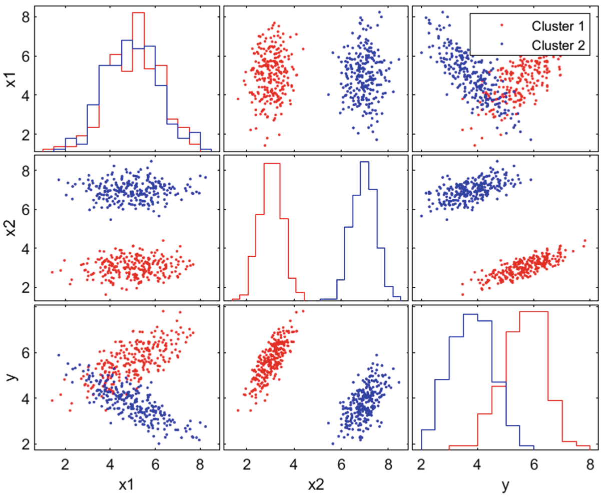


Fig. 2. Plots for the synthetic data set. On the diagonal, the distributions of the variables per cluster are visualized. Each off-diagonal plot is a scatter plot of a column of the data against another column of data.

5 Results

We use two data sets to show the effects of different threshold levels on the estimated fuzzy models. The first data set is synthetic and follows the same distributions as the data set described in [11], the second data set was downloaded from the UCI repository [22].

5.1 Example Case: Synthetic Data Set

For the first tests, we created a data set which contains 500 points and two variables (x_1 and x_2). In the data set there are two clusters, each containing 250 data points. For both clusters, variable x_1 follows a normal distribution $N(\mu, \sigma^2)$ with $\mu = 5$ and $\sigma = 1.2$. For cluster 1, variable x_2 follows the distribution $N(3, 0.5^2)$ and for cluster 2 the values for variable x_2 are drawn from $N(7, 0.5^2)$. The values for the output variable were calculated as $0.4 * x_1 + 1.2 * x_2 + \varepsilon$ for the first cluster and $-0.2 * x_1 + 0.9 * x_2 + \varepsilon$ for the second one. ε is random noise drawn from $N(0, 0.1^2)$. In Fig. 2 a matrix of scatter plots of the input and output variables, and (on the diagonal) frequency histograms of the data is shown.

Using pyFUME, we train first-order Takagi-Sugeno fuzzy models with two rules (and therefore two clusters) for this data set. Therefore, each input variable has two fuzzy sets. pyFUME's similarity threshold is varied from 0.0 to 1.0 in steps of 0.05, and for each level 100 models are built using 75% of the data as training data. All models are then evaluated in terms of Root Mean Square Error (RMSE) with the remaining 25% of the data.

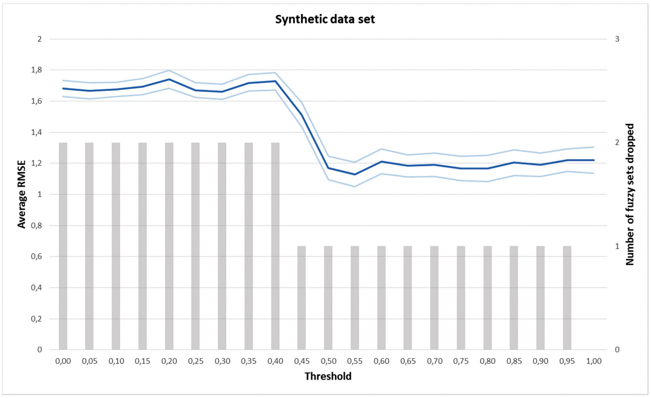


Fig. 3. The average and 95% confidence intervals of the RMSE for fuzzy models for the synthetic data set, build in pyFUME using different similarity thresholds (100 runs each).

In Fig. 3 the average RMSE and the 95% confidence interval for each threshold level are plotted. It can be observed that using very low values for the threshold results in worse performing models. When a similarity threshold of <0.45 is chosen, the two fuzzy sets for each of the variables are deemed similar, and therefore, one of them is dropped. Because of this, only one fuzzy set per variable remains, making both rules identical. As a result, the model does not separate the clusters anymore and behaves like a multiple regression model. This leads to a loss in accuracy.

Using any threshold level $\geqslant 0.45$ but <1.0 results in the merging of the two fuzzy sets for variable $x1$, since these fuzzy sets have a Jaccard similarity index of 0.97. Dropping one of these fuzzy sets does not result in loss of information, since variable $x1$ follows the same distribution in both cluster 1 and 2. Because of this, the RMSE does not decrease when one of the two fuzzy sets is dropped. This can be observed in Fig. 3.

In Fig. 4 the membership function of the fuzzy model that still contains all fuzzy sets is depicted. For variable x_1 indeed a large overlap can be observed for the fuzzy sets. Figure 5 shows the new membership functions for the fuzzy model when the similarity threshold is set to 0.75. Note that for variable x_1 only one set is left. In pyFUME the rules are now simplified to (bold highlights the change):

- `RULE1 = IF (x1 IS cluster1) AND (x2 IS cluster1) THEN (OUTPUT IS fun1)`
- `RULE2 = IF (x1 IS` **`cluster1`**`) AND (x2 IS cluster2) THEN (OUTPUT IS fun2)`

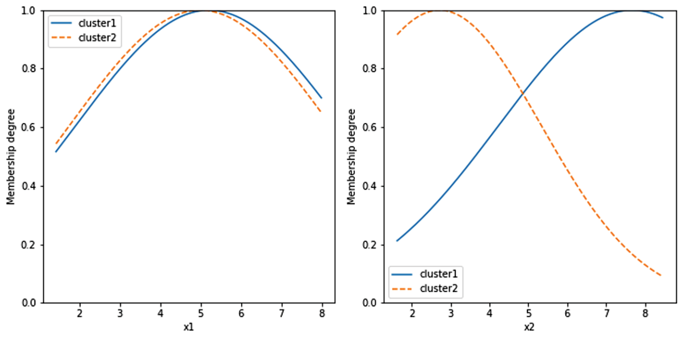


Fig. 4. The membership functions of a fuzzy model based on the synthetic data set. The similarity threshold was set to 1.0 and therefore, no fuzzy sets were dropped.

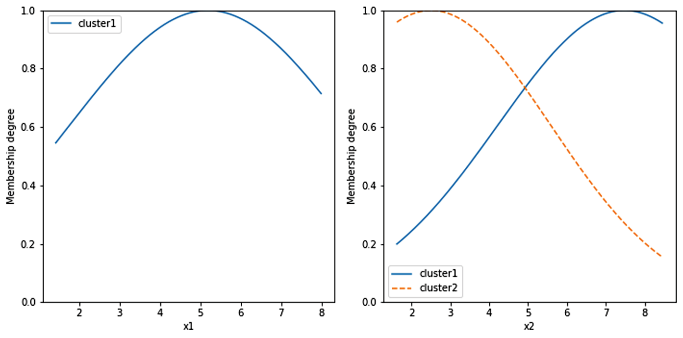


Fig. 5. The membership functions of a fuzzy model based on the synthetic data set. The similarity threshold was set to 0.75 and therefore, one of the fuzzy sets of the variable x_1 was dropped.

5.2 Example Case: NASA Data Set

The NASA data set [5] (downloaded from and described in the UCI repository [22]) consists of 1503 cases of different size NACA 0012 airfoils, which are airfoil shapes for aircraft wings developed by the National Advisory Committee for Aeronautics (NACA). Measurement were taken at various wind tunnel speeds and angles of attack. The span of the airfoil and the position of the observer were kept the same during data gathering in all of the experiments. During the experiments the frequency, angle of attack, chord length, free-stream velocity, and suction side displacement thickness of the NACA 0012 airfoils were recorded. These input variables should be mapped to the output variable, which is the scaled sound pressure level in decibels.

Again, the fuzzy models are build in pyFUME, and a 25% hold-out set is used for testing. To determine the similarity threshold value, different thresholds are tested. The results of this are plotted in Fig. 6 When exploring the effect of these different similarity thresholds on the accuracy of the fuzzy model, it can be observed that dropping fuzzy sets that show a similarity of more than 0.55 to another set does not result in significantly higher error rates. At this threshold, ten fuzzy sets are dropped.

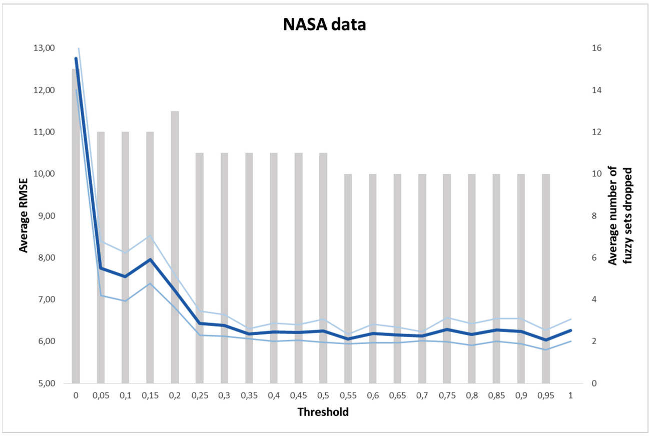


Fig. 6. The average and 95% confidence intervals of the RMSE for fuzzy models for the NASA data set, build in pyFUME using different similarity thresholds (100 runs each).

Figure 7 visualizes the membership functions of the NASA model before fuzzy sets were dropped. In this figure it can be observed that, except for the first variable, all variables have fuzzy sets that show high similarity. These fuzzy sets are dropped when the similarity threshold is set to 0.55, as can be seen in Fig. 8. This figure also shows that the variable 'chord_length' and (to a lesser extent) 'freestream_velocity' have membership functions that are similar to the universal set. This might indicate that these variables can be removed from the model, but that goes beyond the scope of this study. Then, the rule base is as follows (bold highlights the changes):

- RULE1 = IF (frequency IS cluster1) AND (angle_of_attack IS cluster1) AND (chord_length IS cluster1) AND (freestream_velocity IS cluster1) AND (suction_side_displacement_thickness IS cluster1) THEN (OUTPUT IS fun1)
- RULE2 = IF (frequency IS cluster2) AND (angle_of_attack IS cluster2) AND (chord_length IS **cluster1)** AND (freestream_velocity IS **cluster1)** AND (suction_side_displacement_thickness IS cluster2) THEN (OUTPUT IS fun2)
- RULE3 = IF (frequency IS cluster3) AND (angle_of_attack IS **cluster2)** AND (chord_length IS **cluster1)** AND (freestream_velocity IS **cluster1)** AND (suction_side_displacement_thickness IS **cluster2)** THEN (OUTPUT IS fun3)
- RULE4 = IF (frequency IS cluster4) AND (angle_of_attack IS **cluster2)** AND (chord_length IS cluster1) AND (freestream_velocity IS cluster1) AND (suction_side_displacement_thickness IS cluster2) THEN (OUTPUT IS fun4)

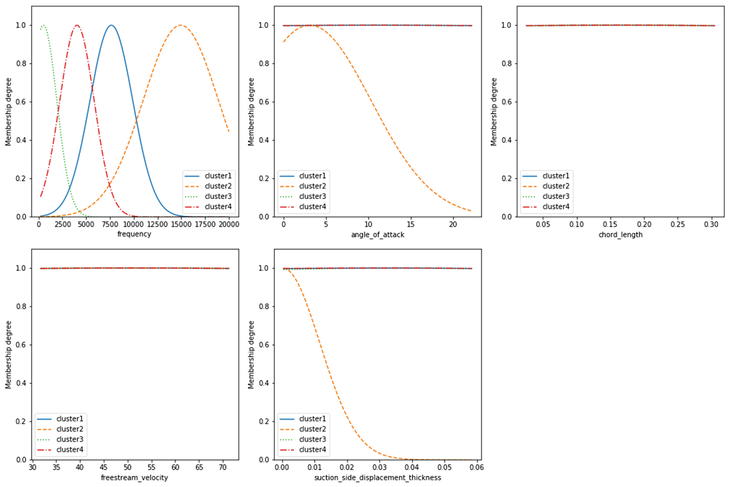


Fig. 7. The membership functions of a fuzzy model based on the NASA data set. The similarity threshold was set to 1.0 and therefore, no fuzzy sets were dropped.

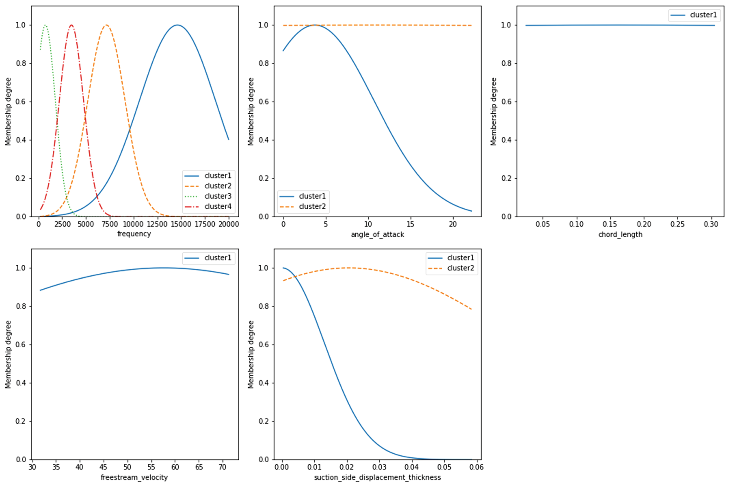


Fig. 8. The membership functions of a fuzzy model based on the NASA data set. The similarity threshold was set to 0.55 and therefore, ten fuzzy sets were dropped.

6 Conclusions

In this paper we introduced a novel graph theory based approach to simplify fuzzy rule bases called GRABS. By combining the Jaccard similarity and graph theory, we determine which fuzzy sets can be simplified in the model. The examples in this paper show that simplifying the model using this approach does not result in significant information and accuracy loss. Future studies will show how these result generalise to other data sets, and how this method compare to other simplification methods. The GRABS approach is implemented in the pyFUME package, whose source code and documentation is available at: https://github.com/CaroFuchs/pyFUME.

In [11] it is shown that when only one fuzzy set remains for a variable, the corresponding antecedent clause can be removed from all the rules in the fuzzy rule base. This improves the readability of the rule base even further. In future releases of pyFUME, we wish to implement the automatic detection and removal of these antecedent clauses. Moreover, we plan to implement a semi-automatic procedure to assign meaningful labels to fuzzy sets (e.g., low, medium, high), after performing simplification, in order to further improve the interpretation of the model. Finally, we will investigate the possibility of exploiting graph measures (e.g., degree centrality) as an alternative to detect fuzzy sets to be removed.

References

1. Alonso, J.M., Castiello, C., Mencar, C.: Interpretability of fuzzy systems: current research trends and prospects. In: Kacprzyk, J., Pedrycz, W. (eds.) Springer Handbook of Computational Intelligence, pp. 219–237. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-43505-2_14
2. Alonso, J.M., Magdalena, L.: Special issue on interpretable fuzzy systems (2011)
3. Babuška, R.: Fuzzy modelling and identification toolbox. Control Engineering Laboratory, Faculty of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands, version 3 (2000)
4. Bezdek, J.C.: Models for pattern recognition. In: Bezdek, J.C., et al. (eds.) Pattern Recognition with Fuzzy Objective Function Algorithms. AAPR, pp. 1–13. Springer, Boston (1981). https://doi.org/10.1007/978-1-4757-0450-1_1
5. Brooks, T.F., Pope, D.S., Marcolini, M.A.: Airfoil self-noise and prediction (1989)
6. Cross, V., Setnes, M.: A study of set-theoretic measures for use with the generalized compatibility-based ranking method. In: 1998 Conference of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No. 98TH8353), pp. 124–129, August 1998. https://doi.org/10.1109/NAFIPS.1998.715549
7. Fan, C.Y., Chang, P.C., Lin, J.J., Hsieh, J.: A hybrid model combining case-based reasoning and fuzzy decision tree for medical data classification. Appl. Soft Comput. **11**(1), 632–644 (2011)
8. Fuchs, C., Spolaor, S., Nobile, M.S., Kaymak, U.: A swarm intelligence approach to avoid local optima in fuzzy c-means clustering. In: 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–6. IEEE (2019)
9. Fuchs, C., Spolaor, S., Nobile, M.S., Kaymak, U.: pyFUME: a Python package for fuzzy model estimation. In: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (2020, accepted)
10. Fuchs, C., Wilbik, A., Kaymak, U.: Towards more specific estimation of membership functions for data-driven fuzzy inference systems. In: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–8. IEEE (2018)
11. Fuchs, C., Wilbik, A., van Loon, S., Boer, A.-K., Kaymak, U.: An enhanced approach to rule base simplification of first-order Takagi-Sugeno fuzzy inference systems. In: Kacprzyk, J., Szmidt, E., Zadrożny, S., Atanassov, K.T., Krawczak, M. (eds.) IWIFSGN/EUSFLAT -2017. AISC, vol. 642, pp. 92–103. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-66824-6_9

12. Guillaume, S.: Designing fuzzy inference systems from data: an interpretability-oriented review. IEEE Trans. Fuzzy Syst. **9**(3), 426–443 (2001)
13. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**(Mar), 1157–1182 (2003)
14. Ho, S.Y., Lee, K.C., Chen, S.S., Ho, S.J.: Accurate modeling and prediction of surface roughness by computer vision in turning operations using an adaptive neuro-fuzzy inference system. Int. J. Mach. Tools Manuf **42**(13), 1441–1446 (2002)
15. Huang, Z.: Rule model simplification. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics (2006)
16. Jaccard, P.: Etude comparative de la distribution florale dans une portion des Alpes et du Jura. Impr, Corbaz (1901)
17. Jin, Y.: Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement. IEEE Trans. Fuzzy Syst. **8**(2), 212–221 (2000)
18. Kaymak, U., Babuska, R.: Compatible cluster merging for fuzzy modelling. In: Proceedings of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium, vol. 2, pp. 897–904. IEEE (1995)
19. Kóczy, L., Hirota, K.: Interpolative reasoning with insufficient evidence in sparse fuzzy rule bases. Inf. Sci. **71**(1–2), 169–201 (1993)
20. Kusiak, A.: Feature transformation methods in data mining. IEEE Trans. Electron. Packag. Manuf. **24**(3), 214–221 (2001)
21. Levrat, E., Voisin, A., Bombardier, S., Brémont, J.: Subjective evaluation of car seat comfort with fuzzy set techniques. Int. J. Intell. Syst. **12**(11–12), 891–913 (1997)
22. Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
23. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Man Mach. Stud. **7**(1), 1–13 (1975)
24. Nobile, M.S., Cazzaniga, P., Besozzi, D., Colombo, R., Mauri, G., Pasi, G.: Fuzzy self-tuning PSO: a settings-free algorithm for global optimization. Swarm Evol. Comput. **39**, 70–85 (2018)
25. Nobile, M.S., et al.: Fuzzy modeling and global optimization to predict novel therapeutic targets in cancer cells. Bioinformatics, btz868 (2019)
26. Oliphant, T.E.: Python for scientific computing. Comput. Sci. Eng. **9**(3), 10–20 (2007)
27. Polat, K., Güneş, S.: An expert system approach based on principal component analysis and adaptive neuro-fuzzy inference system to diagnosis of diabetes disease. Digital Sig. Proc. **17**(4), 702–710 (2007)
28. Raju, G., Zhou, J., Kisner, R.A.: Hierarchical fuzzy control. Int. J. Control **54**(5), 1201–1216 (1991)
29. Setnes, M., Babuska, R., Kaymak, U., van Nauta Lemke, H.R.: Similarity measures in fuzzy rule base simplification. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **28**(3), 376–386 (1998)
30. Shahin, M., Tollner, E., McClendon, R.: Ae-automation and emerging technologies: artificial intelligence classifiers for sorting apples based on watercore. J. Agric. Eng. Res. **79**(3), 265–274 (2001)
31. Spolaor, S., Fuchs, C., Cazzaniga, P., Kaymak, U., Besozzi, D., Nobile, M.S.: Simpful: a user-friendly python library for fuzzy logic (2020, submitted)
32. Sugeno, M., Griffin, M., Bastian, A.: Fuzzy hierarchical control of an unmanned helicopter. In: 17th IFSA World Congress, pp. 179–182 (1993)

33. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **1**, 116–132 (1985)
34. Xiong, N., Litz, L.: Reduction of fuzzy control rules by means of premise learning - method and case study. Fuzzy Sets Syst. **132**(2), 217–231 (2002)
35. Yam, Y., Baranyi, P., Yang, C.T.: Reduction of fuzzy rule base via singular value decomposition. IEEE Trans. Fuzzy Syst. **7**(2), 120–132 (1999)
36. Yen, J., Wang, L.: Simplifying fuzzy rule-based models using orthogonal transformation methods. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **29**(1), 13–24 (1999)
37. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)