Development of a GUI Tool for FML-based Fuzzy System Modeling

Yuichi Omozaki[1], Naoki Masuyama[1], Yusuke Nojima[1], and Hisao Ishibuchi[2*]

[1] Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Osaka, Japan
{yuichi.omozaki@ci.,masuyama@,nojima@}cs.osakafu-u.ac.jp
[2] Shenzhen Key Laboratory of Computational Intelligence, Southern University of Science and Technology, Shenzhen, China
hisao@sustech.edu.cn

Abstract. Fuzzy Markup Language (FML) is a W3C XML-based language defined by the IEEE Standard 1855-2016. FML aims to avoid rewriting or replicating codes developed by different users. An open-source library called Java FML (JFML) is provided to handle FML-based fuzzy system modeling. In JFML, users can manually define various fuzzy sets. However, users are required to have expertise in JFML. Therefore, we develop a GUI tool to design fuzzy sets without any expertise in JFML. Since the tool can output an XML file representing the defined fuzzy sets, they can be applied to any FML-based fuzzy system. We implement an automatic fuzzy partitioning method into the tool. It can also design a simple fuzzy classifier to examine the effect of the defined fuzzy sets on the classification performance. For demonstration purpose, we design several fuzzy classifiers with different membership functions and compare them in terms of the classification performance.

Keywords: Fuzzy Markup Language · The IEEE Standard 1855-2016 · Fuzzy Systems · Software Tool.

1 Introduction

Fuzzy systems have been actively studied thanks to their high interpretability [1]. Fuzzy Markup Language (FML) [2] is the first IEEE standard (1855-2016) for the fuzzy systems. This standard aims to develop fuzzy systems regardless of software environments. In order to provide an operative environment for FML-based fuzzy system modeling, an open-source library called Java FML (JFML) is provided at GitHub [3, 4]. FML-based fuzzy systems are composed of two components called the knowledge base and the rule base. Fuzzy sets and a set of fuzzy rules are defined in the knowledge base and the rule base, respectively. Especially in the knowledge base, users can define, with JFML, membership functions easily by a few parameters (e.g., the shape name and vertexes of the membership functions). However, users need to

* Corresponding Author: Hisao Ishibuchi (hisao@sustech.edu.cn)

have expertise in JFML (i.e., how each parameter affects the shape) to handle FML-based fuzzy systems. Defining various shapes of fuzzy sets should be considered because appropriate fuzzy sets depend on the problem domain and/or user preference. In order to handle FML-based fuzzy systems without expertise in JFML, we develop a GUI tool to easily design FML-based membership functions in this paper. The remainder of this paper is organized as follows. In Section 2, we introduce the GUI tool and explain FML-based membership functions. In Section 3, we compare the performance of several fuzzy classifiers composed of fuzzy sets defined in several ways. Finally, we conclude this paper in Section 4.

2 GUI Tool for Fuzzy Markup Language

2.1 Overview of Our GUI Tool

We develop a GUI tool to handle FML-based membership functions. Fig. 1 shows an overview of the developed tool for FML-based studies/applications. The GUI tool has four functions: 1) an import/export function of an XML file representing membership functions, 2) a visualization function of membership functions, 3) a manual definition and automatic specification function of membership functions, 4) a simple classifier design function to generate a fuzzy classifier with fuzzy if-then rules.

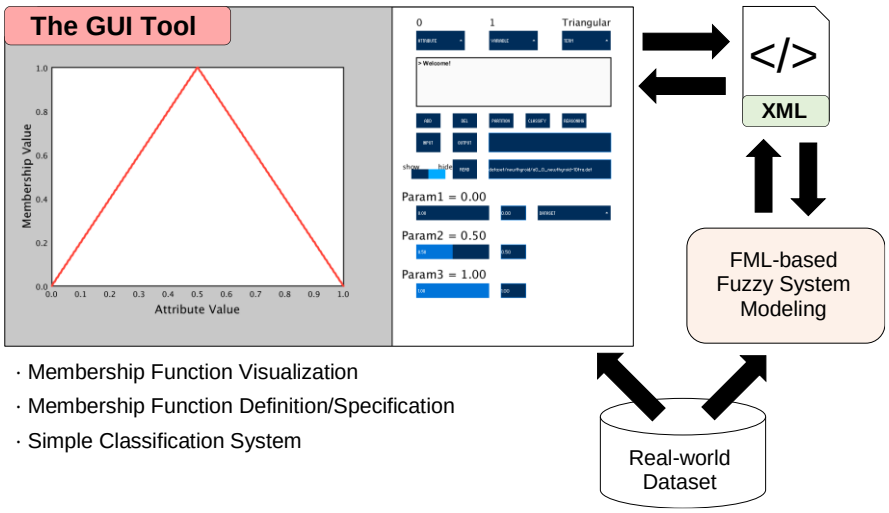


Fig. 1. Overview of the use cases in the field of FML-based studies/applications.

Since the GUI tool can export membership functions as an XML file, users can apply the membership functions to their FML-based fuzzy system modeling. The tool can also import and visualize membership functions learned or obtained in any FML-based fuzzy system modeling. Users can use automatically specified membership functions as a reference. They can modify those membership functions subjectively. Besides, users can check the classification performance of a simple fuzzy classifier composed of fuzzy sets defined by the GUI tool. This quick check helps users to confirm its appropriateness for a dataset at hand. This GUI tool is available from our GitHub repository (https://github.com/CI-labo-OPU/GUI_FMLtool.git).

2.2 Import/Export of FML-based Membership Functions

A fuzzy set *A* is represented by a membership function. Homogeneous triangular-shaped fuzzy partitions are often used in the field of fuzzy systems. Fig. 2 shows the triangular-shaped fuzzy partitions with different granularities. *K* is a given granularity.

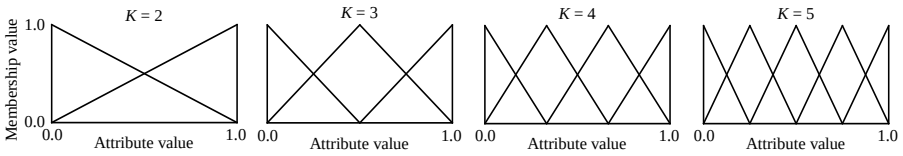


Fig. 2. Homogeneous triangular-shaped fuzzy partitions.

As an example, three triangular-shaped membership functions in the case of $K = 3$ are written in the FML style as follows:

```
<knowledgeBase>
  <fuzzyVariable name="0" type="input">
    <fuzzyTerm name="small">
      <triangularShape param1="0.0" param2="0.0" param3="0.5"/>
    </fuzzyTerm>
    <fuzzyTerm name="medium">
      <triangularShape param1="0.0" param2="0.5" param3="1.0"/>
    </fuzzyTerm>
    <fuzzyTerm name="large">
      <triangularShape param1="0.5" param2="1.0" param3="1.0"/>
    </fuzzyTerm>
  </fuzzyVariable>
</knowledgeBase>
```

JFML requires two elements to define a membership function. One is the name of the shape. The other is a set of a few parameters. For example, the above fuzzy set named "medium" has the name of the shape and three parameters. The name of the shape is defined as "*triangularShape*." The three parameters are defined as the three vertexes of the triangle. JFML also provides other types of shapes. For example, Gaussian-shaped membership functions are also implemented. A Gaussian-shaped membership function has the shape name "*gaussianShape*" and two parameters (i.e., the mean and the variance). In this way, in JFML, the two elements to define a membership function are different by the kind of the shape. For that reason, to design various membership functions, beginners/practitioners of JFML have to understand the parameters.

The GUI tool can import and export an XML file. The imported XML file can be visualized and manually modified on the tool. The exported file can be applied to any FML-based fuzzy systems.

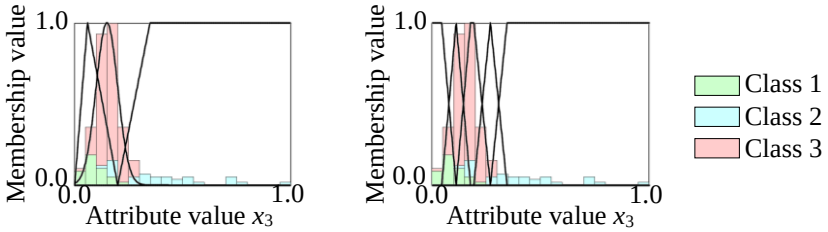
2.3 Manual Definition and Automatic Specification of Membership Functions

Users can define FML-based membership functions by our GUI tool without expertise in JFML. Our tool provides a list to select a kind of membership function shapes. It also provides a few sliders to define parameters of the selected shape. Users can define membership functions easily and subjectively by directly manipulating the list

and the sliders. Since the change is reflected online, users can learn how each parameter affects the shape of membership functions. Users can also subjectively define the membership functions while considering the distribution of the dataset because our tool can show the distribution of the input dataset as the histogram together with membership functions.

On the other hand, in order to specify membership functions automatically from the input dataset, we implement an entropy-based partitioning method for designing inhomogeneous fuzzy sets [5] into our GUI tool. This method was proposed to improve the generalization ability of a fuzzy classifier for test patterns. This method tries to find the partitions that reduce the class entropy of the input dataset.

Fig. 3 shows two examples of the membership functions defined for attribute x_3 in the Newthyroid dataset [6]. Fig. 3 (a) shows the membership functions we subjectively defined. Fig. 3 (b) shows the membership functions automatically specified by the entropy-based method given as $K = 5$.



(a) Subjectively (b) Auto Defined ($K = 5$)

Fig. 3. Two examples of membership functions with the GUI tool for the Newthyroid dataset.

2.4 Simple Fuzzy Classifier Design

Let us assume that we have an n-dimensional M-class pattern classification problem. We have m labeled patterns $\mathbf{x} = (x_1, \ldots, x_n)$ from M classes as training data. The simple fuzzy classifier implemented into the GUI tool is composed of a set of the following fuzzy if-then rules.

$$\text{Rule } R_q\text{: If } x_1 \text{ is } A_{q1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{qn} \quad \text{then Class } C_q \text{ with } CF_q,\ q = 1, 2, \ldots, N, \tag{1}$$

where R_q is the label of the q-th rule, $\mathbf{A}_q = (A_{q1}, \ldots, A_{qn})$ is a set of a q-th antecedent fuzzy set, C_q is a consequent class, CF_q is a rule weight, and N is the number of fuzzy rules. The consequent class C_q and the rule weight CF_q are determined from the training data [7]. The simple fuzzy classifier is composed of all combinations of the antecedent fuzzy sets (i.e., defining five fuzzy sets for each attribute in an n-dimensional pattern classification problem, the number of rules is 5^n). The classifier takes a single winner rule method for reasoning an unseen pattern [7]. The single winner rule is chosen by the maximum value calculated as follows.

$$\textit{Winner } R_q = \max_j \left\{ \mu_{\mathbf{A}_j}(\mathbf{y}) \cdot CF_j \,\middle|\, R_j \in S \right\}, \tag{2}$$

where S is a set of rules as a fuzzy classifier, and $\mu_{\mathbf{A}q}(\cdot)$ is the compatibility grade between an antecedent fuzzy set $\mathbf{A}_q$ and an unseen pattern $\mathbf{y}$. $\mu_{\mathbf{A}q}(\cdot)$ is calculated by the product of the membership values for all attributes. Finally, the consequent class of the single winner rule is output as the predicted class for an input pattern.

3 Computational Experiments

3.1 Experimental Setting

We compare five types of fuzzy classifiers composed of different shapes of fuzzy sets. Three of them use homogeneous fuzzy sets. We use three kinds of homogeneous fuzzy sets, triangular "Tri", Gaussian "Gauss", and trapezoidal "Trape". In addition, the input domain divided into 2-5 partitions. Another type of fuzzy classifier uses the automatically specified fuzzy sets by the GUI tool (referred to as "Auto Def" in Table 1). The last one uses the subjectively defined fuzzy sets (referred to as "By Hand" in Table 1). In this paper, the fuzzy classifiers are composed of all the combinations of the antecedent fuzzy sets. All the used fuzzy sets in the experiment in this paper can be found in our GitHub repository.

We use three real-world datasets, namely Iris (135 patterns, 4 attributes, 3 classes), Newthyroid (193 patterns, 5 attributes, 3 classes), and Phoneme (4,863 patterns, 5 attributes, 2 classes) datasets, which are taken from the UCI Machine Learning Repository [6].

3.2 Experimental Results

The average error rate and the average number of rules over 30 runs (i.e., 10-fold cross-validation × 3) for each dataset are shown in Table 1.

Table 1. Error rates for training/test dataset and numbers of rules.

Dataset		Iris			Newthyroid			Phoneme		
Type	*K*	Train (%)	Test (%)	# of rules	Train (%)	Test (%)	# of rules	Train (%)	Test (%)	# of rules
Tri	5	3.33	4.44	625.0	6.41	7.42	3,125.0	20.48	21.14	3,125.0
	4	3.48	5.33	256.0	8.60	8.82	1,024.0	21.74	22.11	1,024.0
	3	6.22	6.44	81.0	14.09	14.07	243.0	28.09	28.11	243.0
	2	34.00	34.00	16.0	22.51	22.75	32.0	29.35	29.34	32.0
Gauss	5	2.74	4.89	625.0	4.91	6.04	3,125.0	17.57	18.89	3,125.0
	4	7.73	12.67	256.0	4.63	6.81	1,024.0	19.96	20.36	1,024.0
	3	9.48	10.67	81.0	9.78	10.68	243.0	28.31	28.61	243.0
	2	26.96	27.33	16.0	16.06	16.42	32.0	24.69	24.80	32.0
Trape	5	3.88	5.56	625.0	3.82	8.69	3,125.0	16.60	17.75	3,125.0
	4	4.64	7.78	256.0	5.06	7.75	1,024.0	18.91	19.82	1,024.0
	3	2.81	3.33	81.0	9.08	9.76	243.0	25.53	25.97	243.0
	2	14.99	15.33	16.0	18.59	18.74	32.0	25.74	25.74	32.0
Auto Def	5	0.69	6.00	616.7	0.15	3.87	2,500.0	14.84	16.86	3,125.0
	4	1.60	5.11	256.0	1.10	3.87	1,024.0	16.41	17.86	1,024.0
	3	3.90	5.56	81.0	2.79	4.49	243.0	19.00	19.75	243.0
	2	20.72	21.11	16.0	6.03	7.91	32.0	25.62	25.70	32.0
By Hand		4.67	5.78	81.0	5.24	6.04	108.0	24.43	24.38	32.0

Since "Tri" (i.e., fuzzy classifiers with homogeneous triangular membership functions) has been often used for fuzzy classifiers, we compared it with each of the other types (i.e., non-triangular fuzzy sets) in terms of the error rate. For the same *K*, the error rate of the other types is highlighted by red font if it was better than that by "Tri". Almost all of the error rates by the other types were smaller than that by "Tri".

The underline represents the best setting for each dataset. The best error rates were obtained by "Auto Def" with $K = 5$ in most cases.

From Table 1, we can observe a clear tradeoff between the error rate and the complexity (i.e., the number of rules in this paper) except for a classifier with two trapezoidal membership functions (i.e., "Trape" with $K = 2$). In general, a fuzzy classifier with high accuracy has a large number of rules for training data. A large number of rules sometimes lead to an overfitting to the training data and the deterioration in the test data accuracy. In this paper, such a negative effect was not observed except for the use of trapezoidal membership functions.

One interesting observation is that the classifier with membership functions defined by hand (i.e., "By Hand") obtained almost the same error rate as that by the classifier with the automatically specified membership functions when comparing between those classifiers with the same complexity. It probably means that the proposed GUI tool can effectively support a user to manually define membership functions based on the distribution of input patterns.

4 Conclusion

We developed a GUI tool to easily handle FML-based fuzzy sets. The tool requires no expertise in JFML. For demonstration purpose, we compared five types of fuzzy classifiers. The experimental results show the strength of inhomogeneous fuzzy membership functions based on the distribution of input patterns.

In future work, we will discuss an automatic selection method of the appropriate type and granularity of membership functions for each attribute in our GUI tool.

Acknowledgments. This research was partially supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP19K12159.

References

1. A. Fernández, V. López, M. J. del Jesus, F. Herrera: Revisiting Evolutionary Fuzzy Systems: Taxonomy, applications, new trends and challenges. Knowledge-Based Systems. vol. 80, 109-121 (2015)
2. IEEE Standard for Fuzzy Markup Language. Standard 1855-2016 (Available: https://standards.ieee.org/findstds/standard/1855-2016.html) (2016)
3. J. M. Soto-Hidalgo, Jose M. Alonso, G. Acampora, and J. Alcala-Fdez: JFML: A Java Library to Design Fuzzy Logic Systems According to the IEEE Std 1855-2016. IEEE Access. vol. 6, 54952-54964 (2018)
4. Java Fuzzy Markup Language, (https://www.uco.es/JFML/)
5. H. Ishibuchi, Y. Nojima: Comparison between Fuzzy and Interval Partitions in Evolutionary Multiobjective Design of Rule-Based Classification Systems. Proceedings of 2005 IEEE International Conference on Fuzzy Systems. 430-435 (2005)
6. D. Dua, C. Graff: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine (http://archive.ics.uci.edu/ml) (2019)
7. H. Ishibuchi, T. Nakashima, M. Nii: Classification and Modeling with Linguistic Information Granules. pp. 13-22. Springer-Verlag Berlin, Heidelberg (2004)