

## Identifying fuzzy systems from numerical data with Xfuzzy

**I. Baturone**

Instituto de Microelectrónica de Sevilla, (IMSE-CNM), y Dept. Electrónica y Electromagnetismo, Univ. de Sevilla (Spain)  
lumi@imse.cnm.es

**F. J. Moreno-Velo**

Dept. Ing. Electrónica, de Lenguajes Informáticos y Automática, Univ. de Huelva (Spain)  
velo@diesia.uhu.es

**A. Gersnoviez**

Instituto de Microelectrónica de Sevilla, (IMSE-CNM), y Dept. Electrónica y Electromagnetismo, Univ. de Sevilla (Spain)  
andres@imse.cnm.es

### Abstract

Extracting fuzzy rule bases from numerical data draws a great interest nowadays. Several algorithms based on grid partitions or clustering have been reported in the literature for this purpose, and several CAD tools have been developed to implement one or other type of techniques. This paper presents the CAD tool *xfdm* that allows applying both types of techniques. In particular, it permits to select among four types of grid-based techniques and five types of clustering algorithms. Advantages of this tool is that it is integrated into the Xfuzzy 3 environment, and, hence, the identified rule base can be described, verified, tuned, simplified, and synthesized with the corresponding tools of Xfuzzy 3.

**Keywords:** Identification, knowledge extraction, clustering, learning techniques.

### 1 Introduction

Techniques to analyze numerical data and obtain significant information are required in many areas such as medicine, engineering, and, in general, the area of knowledge discovery in databases (KDD). Fuzzy logic-based techniques draw a great interest in these areas since they are able to extract linguistic knowledge that could be easily understood by even non expert people. The reason is that a fuzzy rule base contains rules whose antecedent and consequent parts are expressed with linguistic terms similar to those employed by natural language.

Two different strategies are usually employed when defining a fuzzy rule base from input/output numerical data: clustering and grid-based fuzzy rule learning. The first one organizes the data into clusters and uses them to create the rules. Each cluster is transformed into a rule by projecting it onto every dimension of the input variables [1-2]. Hence, the whole rule base description (rules and membership functions) is obtained simultaneously. An advantage of these techniques is that the number of rules extracted is usually low. As a drawback, each rule has its own fuzzy sets that do not appear in other rules, which complicates their linguistic meaning.

In the other side, the grid-based fuzzy rule learning generates a partition or grid of the input and output spaces prior to creating the rule base [3]. The rules are obtained by selecting the adequate input and output labels according to the numerical data. The grid-based fuzzy rule learning can be understood as a particular case of the clustering technique in which the data are grouped into hyper rectangular clusters formed by the grid partition [4].

Several CAD tools reported in the literature are based on the grid-based fuzzy rule learning technique. Some examples are NEFCLASS [5], NEFPROX [6], and KBCT [7]. Other tools, such as FCLUSTER [8], are based on the cluster-based techniques. Many of these tools are specialized on knowledge extraction problems so that the use of the extracted fuzzy system in a given application must be done with other tools, such as those in Matlab.

The tool presented in this paper, *xfdm*, can apply grid- as well as cluster-based techniques, since it allows using well-known algorithms of both types. In addition, it is integrated within the Xfuzzy 3

environment [9], so that the fuzzy rule base identified can be used by the rest of the tools of this environment without requiring any translation.

The paper is structured as follows: Section II summarizes the design flow that can be followed with the aid of the Xfuzzy environment. It clarifies how this new tool can be used within a fuzzy system design flow. Section III describes the features of this new tool while an illustrative application example is included in Section IV. Finally, some conclusions are given in Section V.

## 2 The Xfuzzy 3 environment

The simplest structure of a fuzzy inference system consists of a unique and simple rule base relating input and output variables with linguistic concepts. The universe of discourse of the variables is covered by fuzzy sets associated with those linguistic concepts. The classic inference mechanism is based on three stages: fuzzification, inference mechanism, and defuzzification. The inference mechanism is performed with three steps: firstly, the activation degree of each rule is calculated by applying conjunctive or disjunctive operators between the antecedents; secondly, an implication function is used to obtain the conclusion of each rule; and, finally, the global conclusion is calculated by an aggregation operator [10].

In order to describe more complex rule bases, complex antecedent parts can be used in the rules, for example, by connecting the several antecedents by any kind of conjunctive and disjunctive connectives, by relating input variables with fuzzy sets by any kind of linguistic hedges, and by even applying linguistic hedges to some connected antecedents [1]. It should be noted that the fuzzy concepts of the rules (such as the membership functions, the conjunctive, disjunctive, implication, and aggregation operators, the linguistic hedges, and the defuzzification methods) can be represented by many mathematical functions and algorithms [11].

In order to describe more complex structures of fuzzy systems, interconnection of several fuzzy modules should be admitted. Each module should apply its own and appropriate fuzzy operators (for instance the defuzzification methods can be different), and may interchange fuzzy or non fuzzy values.

The Xfuzzy 3 environment has been developed so as to be able to design complex fuzzy systems with these three features of expressiveness, extensiveness, and modularity. The design methodology with Xfuzzy follows the flow chart in Fig. 1. The aim of the first stage (description) is to describe the whole fuzzy system, whose constituent modules usually respond to the architecture thought by a human expert. The structure of each module can be also determined from the expert linguistic knowledge. A formal specification language, named XFL3 [12], has been defined to facilitate the translation of complex rules expressed linguistically (with linguistic hedges, weights, any kind of connective function, etc.). The use of XFL3 allows sharing the same system definition throughout the stages of the design methodology. Another relevant feature of the language XFL3 is that it distinguishes between the logical and functional structure of the system to define. The logical structure contains the modular architecture of the system, the linguistic variables employed and their associated fuzzy sets, and the rule base of each module with its associated fuzzy operators. On the other hand, the functional structure contains the mathematical functions which define the membership functions of the fuzzy sets and the fuzzy operators of each module rule base. These operators can be defined freely by the user. The tools *xfedit* and *xfpkg* can aid the user at this description stage.

Once the whole system has been described, its behavior should be tested at the verification stage. Possible errors and deviations have to be detected at this stage in order to correct them. Three tools facilitate this verification process in several ways.

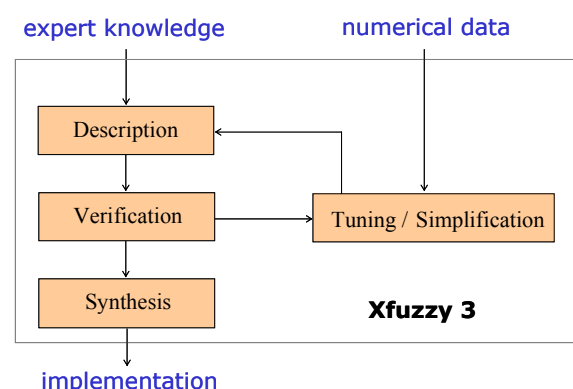


Figure 1: Xfuzzy 3 design flow without identification stage.

One of them (*xfplot*) allows showing two- and three-dimensional graphics with the input/output behavior of the fuzzy system. Another tool (*xfmt*) allows monitoring the values of the internal and global variables of the system and the activation degrees of the rules of the different modules. The last tool, named *xfsim*, simulates the behavior of the fuzzy system working in line with a model of an external system (a plant in the case of a controller).

One way of correcting deviations from a desired behavior is to apply supervised learning techniques. The tool *xfsl* includes a wide set of tuning algorithms [13]. It allows tuning hierarchical fuzzy systems with operators defined freely by the user. After tuning, the resulting module descriptions might be simplified by applying pruning techniques. This is performed by the tool *xfsp*.

Once the whole system description has been verified, tuned, and simplified, the last step (synthesis) is to represent the system by an executable format suitable for the application. Three tools allow software synthesis (to C, C++, and Java) while the tool *xfvhdl* allows hardware synthesis.

This design flow starts with fuzzy rule bases translated from linguistic knowledge, but it did not allow obtaining rule bases from numerical data. The numerical data were employed only to adjust or tune the fine structure of the rule bases (the parameters of the membership functions). The new tool, *xfdm*, has been developed to also employ these numerical data to obtain the coarse structure of the rule base (number of membership functions, number of rules, etc.). Hence, a new identification stage has been included within this design flow, as shown in Fig. 2.

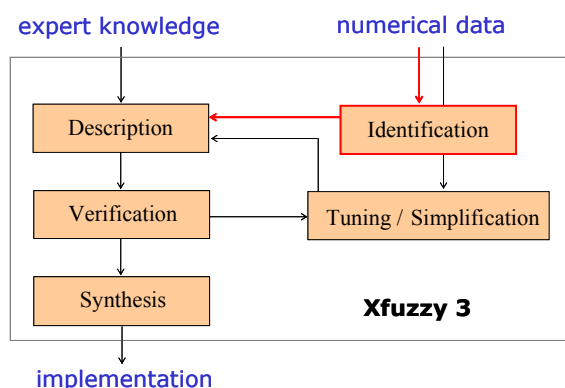


Figure 2: Xfuzzy design flow with identification stage.

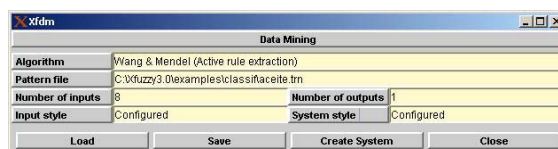


Figure 3: Main window of the *xfdm* tool.

### 3 The *xfdm* tool

The main window of the *xfdm* tool is shown in Fig. 3. It allows selecting: the grid- or clustering-based algorithm employed to extract the fuzzy rule base, the file with the numerical data from which the fuzzy rule base will be extracted, the number of inputs and outputs of the rule base to extract, and the input and system style of that rule base. The input style means the number and type of membership functions to cover the input universes of discourse as well as the name of the inputs. It is configured with the window shown in Fig. 4. Depending on the algorithm, this window will allow or not selecting different number and types of membership functions for each input variable. Free triangles or Gaussian functions as well as families of triangles and B-splines can be selected to represent the input fuzzy sets, as shown in Fig. 4. Another window, shown in Fig. 5, helps the user to configure the system style. The system style means to specify the name of the rule base to extract, the prefix used to name the output variables, the kind of conjunction operator used in the antecedents, and the type of inference-defuzzification method applied (Fuzzy Mean, Weighted Fuzzy Mean, Takagi-Sugeno, or a classification method, which takes as output the consequent of the most activated rule). Regarding

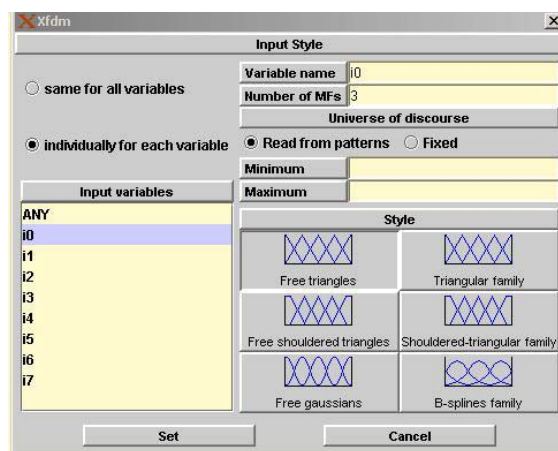


Figure 4: Window to configure the input style.





