FCA-Based Rule Generator: a framework for the genetic generation of fuzzy classification systems using formal concept analysis

http://www.producao.usp.br/handle/BDPI/50002

Downloaded from: Biblioteca Digital da Produção Intelectual - BDPI, Universidade de São Paulo
FeA-BASED RULE GENERATOR, a framework for the genetic generation of fuzzy classification systems using formal concept analysis

Marcos E. Cintra
Natural and Exact Sciences Department
Federal Territorial Uni. of the Semi-Arid, Brazil
{mecintra@gmail.com}

Maria C. Monard
Inst. of Math. and Computer Sciences
Sao Paulo University, Brazil
Email: mcmonard@icmc.usp.br

Heloisa A. Camargo
Computer Science Department
Federal Uni. of Sao Carlos, Brazil
Email: heloisa@dc.ufscar.br

Abstract—There is a number of frameworks for the general task of classification available for free usage on the Internet. However, software to generate fuzzy classification systems using the genetic approach is scarce. In this work, we present the FCA-BASED RULE GENERATOR framework to automatically generate fuzzy classification systems based on a genetic rule selection process. Such rules are extracted from data using a formal concept analysis approach. The FCA-BASED RULE GENERATOR framework includes modules for dataset preprocessing, automatic definition of fuzzy data bases from data, dataset optimization, a module based on formal concept analysis for rule extraction, a genetic algorithm module, as well as a rule base optimization module. The software is described and an example of use is presented.

1. INTRODUCTION

The task of classification has been widely researched by both, the machine learning and the soft computing communities. The classification methods proposed in the literature include different approaches, such as artificial neural networks [1], support vector machines [2], as well as rule-based approaches, including decision trees [3], [4] and Genetic Fuzzy Systems [5], [6], [7], [8]. Implementations of such proposals can be found in different frameworks for public use.

Fuzzy systems, the focus of this work, are composed of: i) a fuzzy rule base, which contains the knowledge of the domain; ii) a fuzzy data base, which contains the definition of the attributes in terms of fuzzy sets, and iii) an inference mechanism. Fuzzy systems can be defined by experts or automatically learned from data.

Genetic fuzzy systems, in turn, are fuzzy systems that combine the global search of Genetic Algorithms (GA) with the fuzzy set theory to provide accurate and interpretable rule-based systems. Genetic fuzzy systems can be classified according to the objective of the GA (learning or adjusting components), according to component(s) to be learned (the fuzzy rule base, fuzzy data base, or both) as well as by the strategy used by the GA (rule selection, rule extraction, multi-objective, among others). A classification of the various genetic fuzzy methods found in the literature according to a comprehensive taxonomy is presented in [9]. The work in [10] reviews the approaches that use Multi-objective genetic Algorithms to generate fuzzy rule based systems from data.

In [11], a method to automatically generate fuzzy rule based systems using the genetic approach with rule selection, namely the FCA-BASED RULE GENERATOR method, is proposed. A predefined fuzzy data base is used, allowing the GA to focus its search exclusively on the generation of suitable fuzzy rule bases. Also, by using a predefined fuzzy data base, it is possible to automatically build and extract rules, i.e., a set of candidate rules that define the genetic search space. Such rules are extracted based on the Formal Concept Analysis (FCA) theory [12], [13]. Another advantage of the FCA-BASED RULE GENERATOR method is that, as the search space is defined before the GA search process, it is possible to use simple and efficient codifications for the chromosomes that improve the use of genetic operators and accelerate the whole genetic process.

As most of the methods required by FCA-BASED RULE GENERATOR are new proposals, a framework was developed to implement them using the Java language. The current version works as a command line application

The FCA-BASED RULE GENERATOR method includes the following desirable characteristics for rule extraction:

- Rules are extracted directly from the data, avoiding the extraction of a potentially large number of unnecessary/useless rules;
- Unlike the exhaustive generation of rules, which has exponential complexity, by using FCA the complexity of the process is typically polynomial [14];
- The extracted rules present variable number of conditions in their antecedents, improving the interpretability of the generated fuzzy rule bases;

In this paper, we present related frameworks and detail the FCA-BASED RULE GENERATOR method, discussing the modules that compose it, as well as their implementation and computational complexity. Notice that most of the proposal was previously described in [15] with a theoretical approach. This paper, on the other hand, aims at describing the implementation of the proposal and its technical details.

The remainder of this paper is organized as follows: Section II discusses related and publicly available software.

1Available at https://dl.dropboxusercontent.com/u/16102646/FCARuleGenerator.zip
Section III details the FCA-BASED RULE GENERATOR framework and its modules. Section IV presents an example of usage, followed by the conclusions and future work in Section V.

II. RELATED WORK

In the literature, it is possible to find different types of software for the task of classification using fuzzy logic, the focus of the FCA-BASED RULE GENERATOR method. Next, we present some of the most relevant ones in regards to the supervised learning, specifically for soft computing.

A. KEEL

Regarding the genetic generation of fuzzy systems, the KEEL (Knowledge Extraction based on Evolutionary Learning) framework [16], [17] is one of the most well-known. KEEL includes several algorithms from the machine learning and soft computing areas. The methods and algorithms available in KEEL which are of interest to the FCA-BASED RULE GENERATOR framework are categorized into:

- Data preprocessing;
- Classification;
- Clustering algorithms;
- Association rules;
- Statistical Tests.

The KEEL framework contains several fuzzy and classic methods in each of the aforementioned categories. KEEL allows the definition of experiments using a graphical interface that is user-friendly and intuitive. Moreover, it allows the combination of multiple algorithms making it quite simple to organize massive experiments. It is developed using the Java language and licensed under the GNU General Public License (GPLv3). KEEL is distributed in English.

B. Xfuzzy

Xfuzzy [18] is a development environment for fuzzy-inference-based systems. It is composed of several tools that cover the different stages of the fuzzy system design process, from their initial description to the final implementation. Its main features are the capability for developing complex systems and the flexibility of allowing the user to extend the set of available functions. Xfuzzy is developed in Java and licensed under the GNU General Public License. It is operating system independent.

Xfuzzy 3.0 is a complete environment for the design of fuzzy systems. Its tools, covering the different stages of the design process, share a common formal description language which allows the definition of complex systems by means of hierarchical rule bases and user-configurable membership functions, defuzzification methods, fuzzy connectives, linguistic hedges and implication and aggregation operators.

The tools in Xfuzzy can be executed independently. It is distributed in English, with manuals available in Spanish and Portuguese as well.

C. GUAJE

GUAJE [19], Generating Understandable and Accurate fuzzy models in a Java Environment, implements a fuzzy modelling methodology named as Highly Interpretable Linguistic Knowledge (HILK), which is aimed at yielding a good interpretability-accuracy trade-off by combining expert and induced knowledge in a common framework. It allows building interpretable and accurate fuzzy systems by means of combining several pre-existing open source tools, taking profit from the main advantages of each individual tool by analogy with the main idea underlying Soft Computing.

GUAJE is user-friendly, portable, and developed in order to make easier knowledge extraction and representation for fuzzy logic based systems. Its main goal is the generation or refinement of fuzzy knowledge bases with a particular interest of obtaining interpretable partitions and rules. GUAJE lets the user define expert variables and rules, but also provide learning capabilities for partitions and rules. Both types of knowledge, expert and induced, are integrated under the expert supervision, ensuring interpretability, simplicity and consistency of the knowledge base along the whole process.

The main algorithms available at GUAJE are:

- Data pre-processing;
- Feature selection;
- Partition design;
- Rule base definition;
- Knowledge base visualization, improvement, and validation.

D. FuzzyStudio

FuzzyStudio [20], is an on-line tool for modelling and simulating fuzzy systems. The tool aims to facilitate the fuzzy systems building process focusing on the learning of fuzzy systems by undergraduate students.

As it is an on-line system, in order to be accessed, one has to register to the system. FuzzyStudio allows the definition of fuzzy classification systems, as well as their executions. Systems can be saved for later usage. It also allows Fuzzy Control Language standard code generation and Java code generation. Furthermore, the tool enables collaborative work, allowing more than one user to simultaneously work on the same project.

FuzzyStudio supports Mamdani-type fuzzy systems [21] and its implementation is exploited with the jFuzzyLogic library. FuzzyStudio is currently only available in Portuguese.

E. Considerations

KEEL is a robust framework that suits scientific experiments due to the possibility of arranging experiments with different methods, covering the stages of preprocessing to comparisons of results using statistical tests, that can be remotely executed using command line controls. Xfuzzy and

2Available at http://keel.es/
3Details at http://www.gnu.org/licenses/quick-guide-gplv3.html
4Available at http://www2.imse-cnm.csic.es/Xfuzzy/
5http://sourceforge.net/p/guajefuzzy/wiki/Home/
6http://bsi.ceavi.udesc.br:8082/FuzzyStudio/
GUAJE focus on the definition and generation of fuzzy models. FuzzyStydio, on the other hand, is a didactic tool which suits best students of fuzzy systems, especially for tasks such as control.

Next, we present the FCA-BASED RULE GENERATOR method and framework.

III. THE FCA-BASED RULE GENERATOR FRAMEWORK

The FCA-BASED RULE GENERATOR method is able to generate fuzzy knowledge bases, i.e., fuzzy data bases and fuzzy rule bases, from data. The FCA-BASED RULE GENERATOR framework is composed of the following modules:

1) Data preprocessing;
2) Data base definition;
3) Dataset example reduction;
4) Fuzzy formal context definition;
5) Fuzzy rule extraction;
6) Genetic selection of rules;
7) Post rule pruning.

The flowchart in Figure 1 shows the operations of the FCA-BASED RULE GENERATOR framework from the preprocessing of a dataset to the definition of the final fuzzy rule base.

For each dataset, FCA-BASED RULE GENERATOR starts by creating n folds of train and test examples. Next, a fuzzy data base is automatically defined. Once the fuzzy data base is generated, a process is applied to remove similar examples in order to improve the whole fuzzy rule base generation process. Using the FCA theory, the datasets are then used to generate fuzzy contexts in order to allow the extraction of fuzzy classification rules. A genetic rule selection process is performed in order to define the fuzzy rule base. Such fuzzy rule bases are then pruned to form the final fuzzy rule base.

FCA-BASED RULE GENERATOR is implemented using Java and distributed under the GNU GPL licence.

Algorithm 1 presents the FCA-BASED RULE GENERATOR method.

Algorithm 1: The FCA-BASED RULE GENERATOR method.

\[\text{Algorithm 1: The FCA-BASED RULE GENERATOR method.}\]

- **input**: A given dataset
- **Define** the fuzzy data base using the FuzzyDBD method;
- **Remove** similar examples [22];
- **Generate** the formal context;
- **Extract** the classification rules from the formal context (see Algorithm 2, page 4);
- **Using** the Wang & Mendell method [23], define the values for the initial number of rules (Best_NR) and correct classification rate (Best_CCR);
- **Create** the initial population of the GA formed by k chromosomes, each with Best_NR genes;
- **for** \( a = 1 \) to the maximum number of iterations **do**
  - Calculate and update the fitness value of the chromosomes (see Algorithm 5, page 6);
  - Apply the elitism, mutation, and crossover genetic operators;
- **Perform** a rule pruning process.

Each of the modules forming the FCA-BASED RULE GENERATOR framework is described next.

A. Data pre-processing module

The main task performed during the preprocessing stage is the generation of \( n \) folds for test and training with balanced examples in regards to the classes of the datasets. Examples with missing values are removed. Outlier processing is not considered.

Regarding the technical issues of this module, as the pairs of train and test examples should be balanced in relation to the number of examples of each class, it is necessary to sort the examples and then split them into different sets.

B. Data Base Definition Module

The FCA-BASED RULE GENERATOR framework has the Equalized Universe Method [24] implementation to define fuzzy data bases. The Equalized Universe Method splits the domains of the attributes of the datasets into even subspaces which are used to define the membership functions. FCA-BASED RULE GENERATOR uses triangular and trapezoidal shaped fuzzy sets.

FCA-BASED RULE GENERATOR also incorporates the FuzzyDBD method [25] to define fuzzy data bases. The FuzzyDBD method uses three distinct estimation functions.
to individually estimate the number of fuzzy sets for each attribute. Its implementation was designed based on the easiness of usage and speed by considering two reasonable simplifications for the process: the number of fuzzy sets for each attribute range from 2 to 9, and it is assumed independence among attributes. The process works similarly to a wrapper to estimate the number of fuzzy sets to each individual attribute. Both methods have linear complexity.

C. Dataset example reduction module

As the cost of extracting rules using FCA-BASED RULE GENERATOR is directly to the number of examples and attributes of the dataset used, an algorithm [22] is used to compare the examples and remove similar ones.

The comparison is performed based on the fuzzyfication of the examples of the dataset. In this sense, the examples have their continuous attribute values fuzzyfied using the fuzzy sets with highest membership degree with it and then each example is compared to all the others. When two or more identical examples are found, only one of them is kept. Experimental results show that the cost of such process is compensated with a substantial reduction in the cost to extract rules using the FCA-BASED RULE GENERATOR method.

D. FCA-Extraction module

The method for rule extraction from data performed by FCA-BASED RULE GENERATOR is directly to the number of examples and attributes of the dataset used, an algorithm [22] is used to compare the examples and remove similar ones.

One of the applications of formal concept analysis is related to the extraction of association and classification rules. For the extraction of classification rules from a formal context, proposed in [11].

Algorithm 2: Extraction of Rules using FCA.

Input: A given dataset described by m attributes and n examples
1. Perform a scaling process, i.e., define the continuous attributes of the dataset in terms of fuzzy sets (see Algorithm 3);
2. Extract the formal concepts, or part of them, from the formal context using the modified version of the NextClosure algorithm (see Algorithm 4);
3. Discard the extension of the extracted formal concepts;
4. Store the formal concepts with a class, forming classification rules.

1) Fuzzy scaling process: The FCA-BASED RULE GENERATOR method contains a fuzzy scaling process, i.e., the transformation of continuous attributes into attributes defined by fuzzy sets. The fuzzy sets defining each attribute are used by the scaling process as binary values. Algorithm 3 details the scaling process.

To guide the scaling of real values, a predefined threshold $A_{min}$ for the membership degrees is used. Whenever the membership degree of a certain input value for a fuzzy set is equal or higher than $A_{min}$, the corresponding attribute is set to true in the formal context, otherwise, the attribute is set to false.
Algorithm 3: Scaling process used by FCA-BASED RULE GENERATOR.

Input: A given dataset described by m attributes and n examples.
1 Define the FDB, i.e., define the continuous attributes in terms of fuzzy sets.

2 for a = 1 to m do
3     if Attribute $A_{att_a}$ is continuous then
4         create a binary attribute for each fuzzy set defining $A_{att_a}$;
5     else create a binary attribute for each discrete value of $A_{att_a}$;
6 Define $A_{min}$ as the minimum membership degree to guide the fuzzy scaling process;
7 for e = 1 to m do
8     for a = 1 to m do
9         if $A_{att_a}$ is a continuous attribute then
10            for x = 1 to total number of binary attributes
11               calculate $A_{Att_a}$ as the membership degree of the input value of attribute $A_{att_a}$ from example e in the fuzzy set defining the binary attribute $A_{att_a}$;
12               if $A_{Att_a} \geq A_{min}$ then
13                   set binary attribute $A_{att_a}$ as true;
14               else set binary attribute $A_{att_a}$ as false;
15         else $A_{att_a}$ is a discrete attribute
16            for x = 1 to total number of label values of attribute $A_{att_a}$ do
17               if x = the input value of attribute $A_{att_a}$ from example e then
18                  set binary attribute $A_{att_a}$ as true;
19               else set binary attribute $A_{att_a}$ as false;

Notice that as different algorithms extract exactly the same sets of formal concepts for a given formal context, step 2 can be performed by any extraction algorithm.

2) Formal concept extraction process: The algorithms available in the literature for the extraction of formal concepts include the NextClosure algorithm, which works by finding neighbouring concepts [26], and the algorithm proposed in [27], which features a parallel search process. NextClosure can be used to extract formal concepts by analysing the attributes or the objects of a formal context. When using the attributes, it is called Next Intent, while it is called Next Extent when used with the objects. We adapted the NextClosure algorithm to extract classification rules, as shown in Algorithm 4.

As the NextClosure algorithm looks for neighbouring concepts, its modified version verifies whether a found concept has a class or not. If it does, then the formal concept can be stored in a data structure including all extracted concepts. Otherwise, the process discards the found formal concept and carries on looking for the next neighbouring concept. This modification is represented by lines 8 and 9 of Algorithm 4.

An important issue regarding the extraction of the formal concepts when using the binary fuzzification of the formal context is related to the increase in the number of attributes. The total number of attributes after the scaling of continuous and multi-valued attributes is equal to the sum of all fuzzy sets describing each original attribute, plus all the values of each discrete attribute, and the number of classes. This way, an increase in the number of fuzzy sets defining the attributes leads to an exponential increase in the number of possible formal concepts.

The total number of formal concepts in a formal context can be estimated using the Metropolis-Hastings algorithm for sampling formal concepts, described in [28]. Nevertheless, it is important to notice that for our purpose, the total number of formal concepts is much larger than the number of formal concepts we are interested in extracting: as we want to extract classification rules, we are only interested in extracting formal concepts that have a class in their intention. This way, it is possible to reduce the number of extracted rules using different methods. Next, we describe the methods implemented to reduce the number of extracted rules by the FCA framework, as well as its parameters and default values.

- Restricting membership degree for scaling of attributes (see Algorithm 3) is set to more than 0.5, less binary attributes are activated, generating less formal concepts. As this parameter is required by the method, it is implemented in the framework.
- Evaluating the support of concepts. The support of a concept is the number of objects it shares divided by the number of all objects. Thus, a minimum support can be set to reduce the number of extracted formal concepts, while considering class distributions. This method is also implemented and available in the FCA-BASED RULE GENERATOR framework.
- Predefining a maximum number of rules. The default value is set to allow the extraction of all possible rules.
- Using probability to extract a percentage of the rules. This option is also implemented using a random variable which is set whenever a rule is extracted. This way, the value of the random variable is compared to the percentage of rules desired in order to keep it or discard it.

Notice that a significant reduction in the time to extract the formal concepts is obtained as a result of the reduction in the examples of the training sets, performed in the Dataset Example Reduction module.

Algorithm 4: Adaptation of the Next Closure Algorithm.

Input: A closure operator $X \rightarrow X^\prime$ on a finite set $M$, and a subset $A \subseteq B$.

Output: $A$ is replaced by the lexicographically next closed set.

1. $i :=$ largest element of $M$;
2. $success := false$;
3. repeat
4.     if $i \notin A$ then
5.         $A := A \cup \{1, 2, ..., (i - 1)\}$;
6.         $B := A^\prime$;
7.     if $(B - A)$ contains no element $< i$ then
8.         if $B$ contains a class then
9.             $A := B$ (Concept $A$ can be extracted to generate a classification rule);
10.        success := true;
11.    else
12.        $A := A - \{i\}$;
13.    $i := pred(i)$; 
14. until success or ($i > (number$ of elements in $M$ - number of classes of the dataset)$);
E. Genetic algorithm module

The main feature of the FCA-BASED RULE GENERATOR method is the extraction of rules using the FCA proposal. Once the rules forming the genetic search space are extracted in the rule extraction module, FCA-BASED RULE GENERATOR selects rules using a genetic algorithm to define fuzzy rule bases. This approach is referred to as rule generation by selection with an a priori rule extraction [9]. Notice that the rules of the genetic search space are predefined and fixed, i.e., the genetic process does not alter the existing rules. Thus, in this kind of approach, the predefined set of rules is usually called the set of candidate rules.

Next, we present the main methods of the genetic module of FCA-BASED RULE GENERATOR.

1) Chromosome Codification: FCA-BASED RULE GENERATOR uses an integer chromosome codification: each chromosome represents a complete rule base. In the integer codification, each gene contains an index to a rule in the search space of candidate rules. The size of the chromosome, i.e., its number of genes, is defined as the number of rules found in the rule base produced by the Wang & Mendell (WM) method [21] applied to a given dataset. By using this reference value, it is possible to work with reasonably small chromosomes. In order to allow the generation of rule bases with less rules than the total number of rules in the WM rule base, a “−1” value is used to indicate that a gene represents an inactive rule.

The integer codification requires considerably less memory than the binary codification. Also, the use of the Wang & Mendell method to define the size of the chromosomes allows an optimization of the chromosomes size and, thus, yields less computational effort and memory requirements.

2) Chromosome Initialization: The chromosomes of the initial populations are randomly generated. The existence of conflicting or redundant rules is verified in the initialization and when genetic operators are applied. To induce the reduction of the number of active rules in the chromosomes, we initialize the chromosomes with different percentages of active rules: 25% of the chromosomes are initialized with all rules activated, 25% with 80% of active rules, 25% with 60% of active rules, and the remaining 25% with 40% of active rules. By initializing 75% of the population with less rules than the maximum possible, we allow the generation of reduced rule bases. The percentages of active rules in the chromosomes were defined empirically.

The chromosome initialization requires a method to check for conflicting rules and redundant rules, as well as a method to create the data structures that define the chromosomes.

3) Fitness Calculation: The fitness value of each chromosome is calculated using the Correct Classification Rate (CCR) and the Number of Rules (NR) in the fuzzy rule base represented by each chromosome, as proposed in [11]. This evaluation process uses a self-adaptive algorithm that updates referential values of ideal accuracy (Best_CCR) and number of rules (Best_NR).

The initial values of Best_CCR and Best_NR are set before the use of the GA, and using the same fuzzy data base. These values are updated after each generation when a better accuracy is obtained by a chromosome in the population. The number of rules is used in a penalization mechanism that decreases the fitness value of a chromosome when its number of rules is larger than the current reference number (Best_NR). Table I presents the penalization rates defined empirically.

<table>
<thead>
<tr>
<th>Number of rules</th>
<th>Penalization rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\leq$ Best_NR</td>
<td>no penalization</td>
</tr>
<tr>
<td>$(\text{Best_NR} \times 1.5)$</td>
<td>10%</td>
</tr>
<tr>
<td>$(\text{Best_NR} \times 2)$</td>
<td>30%</td>
</tr>
<tr>
<td>$(\text{Best_NR} \times 3)$</td>
<td>40%</td>
</tr>
<tr>
<td>$&gt;(\text{Best_NR} \times 3)$</td>
<td>50%</td>
</tr>
</tbody>
</table>

Algorithm 5 presents the fitness calculation procedure for the FCA-BASED RULE GENERATOR method. Aiming at avoiding a premature convergence of the GA, FCA-BASED RULE GENERATOR uses a Self-Adaptive Genetic Algorithm (SAGA) [29] that dynamically adjusts the crossover and mutation rates at each new generation.

Algorithm 5: FCA-BASED RULE GENERATOR fitness calculation.

```
input: The population of the GA formed by $k$ chromosomes
input: Initial number of rules (Best_NR)

1 for $c = 1$ to $k$ do
2     $f_c = CCR$, the initial fitness of chromosome $c$;
3     if $f_c > \text{Best\_CCR}$ then
4         $\text{Best\_CCR} = f_c$;
5         $\text{Best\_NR} = \text{number of rules in chromosome } c$;
6     if $f_c = \text{Best\_CCR}$ then
7         if number of rules in chromosome $c < \text{Best\_NR}$ then
8             $\text{Best\_NR} = \text{number of rules in chromosome } c$;
9     for $c = 1$ to $k$ do
10        if $f_c < \text{Best\_CCR}$ then
11            Apply a penalization to $f_c$ according to Table I;
```

The implementation of the methods that perform the fitness calculation is based on the optimization of the whole genetic process. In this sense, by allowing a variation in the rates of crossover and mutation, but, at the same time, penalizing the solutions based on the best ones found in the population, it is possible to speed up the genetic process while maintaining its global search characteristic.

F. Post Rule Pruning Module

In order to improve the interpretability of the final rule bases it generates, FCA-BASED RULE GENERATOR performs a post selection process. This rule selection process checks the ability of each individual rule to improve the classification power of the rule set.

In this sense, this method simply remove rules individually and calculates the accuracy rate of the fuzzy rule base. If the accuracy improves, the rule is removed permanently, otherwise, it is kept in the fuzzy rule base and the process checks the next rule. Notice that this rule pruning strategy is sensitive
to the order rules are presented. Notice this method is sensitive to the order of the rules.

In terms of implementation, most of the computational cost of this procedure is taken by the classification. Nevertheless, as the number of rules is limited and optimized by the fitness procedure, the computational cost of this method is minimized.

G. Fuzzy Classification Module

Although the fuzzy classifier is not described as a module of the FCA-BASED RULE GENERATOR method, it is used in many parts of the FCA-BASED RULE GENERATOR method. For instance, the fuzzy classification module is used by the fuzzy database definition module, it is also used by the genetic algorithm module to calculate the fitness of chromosomes during the genetic selection of rules, as well as by the rule pruning module.

The implemented classifier uses the Mamdani approach [21]. This way, the minimum t-norm is used for the conjunction of the antecedents and the maximum s-norm for the aggregation of the rules.

Regarding the inference mechanisms, the classic and the general fuzzy reasoning methods are implemented [30]. This way, an input example can be classified with the class of the rule with highest compatibility degree with it (classic reasoning) or with the class with the highest compatibility degree sum (general reasoning), considering all rules from the fuzzy rule base.

IV. EXAMPLE OF USAGE

FCA-BASED RULE GENERATOR receives an attribute relation file format (.arff) 7 file as input and generates the following files:

- A set of $K$ rule base files, one for each fold (named “dataset”BestRuleBaseTrainSet$K$.txt);
- A data base file (named “dataset”DataBase.txt);
- A file with all rules extracted using the FCA approach (named “dataset”-FCAExtractedRules.txt);

The input parameters, and default values in brackets, for FCA-BASED RULE GENERATOR are:

- Dataset name;
- Number of sets defining each attribute (3 sets);
- Inference method, classic or general (classic);
- Percentage of rules to be extracted using the FCA approach (100%);
- Number of chromosomes (100);
- Number of generations (200);
- Mutation rate (5%);
- Crossover rate (70%);
- Elitism rate (10%);
- Minimum support for the extraction of rules using the FCA approach (0);
- Number of folds (10);

The dataset name is the only required parameter. If any of the other parameters is not defined, the default values are used.

Next, we present some examples of execution of FCA-BASED RULE GENERATOR using the Iris dataset from the UCI Machine learning repository [31]:

- Using default parameters:
  – java -jar FuzzyFCA.jar iris
- Using custom parameters:
  – java -jar FuzzyFCA.jar iris 3 classic 100 50 100 0.05 0.7 0.1 0 10 (3 fuzzy sets, classic inference method, extracting 100% of the rules using FCA, 50 chromosomes, 100 generations, 5% mutation rate, 70% crossover rate, 10% elitism rate, no minimum support for extracted rules, 10 folds);
  – java -jar FuzzyFCA.jar iris 5 general 50 100 200 0.05 0.7 0.1 0 10 (5 fuzzy sets, general inference method, extracting 50% of the rules using FCA);
  – java -jar FuzzyFCA.jar iris 7 classic 100 200 1000 (using default values for mutation, crossover, and elitism rates, no minimum support and 10 folds);

Next, we present two fuzzy rule bases generated by FCA-BASED RULE GENERATOR for the well-known iris dataset [31] is, the first with three rules, and the second one with four rules. These rule bases were generated using default parameter values.

1) If SepalLength is Small and SepalWidth is Small and PetalLength is Medium and PetalWidth is Medium then class is versicolor
2) If SepalLength is Small and PetalLength is Small and PetalWidth is Small then class is setosa
3) If PetalLength is Large and PetalWidth is Large then class is virginica

1) If PetalLength is Small and PetalWidth is Small then class is versicolor
2) If SepalLength is Medium and SepalWidth is Medium and PetalWidth is Medium then class is virginica
3) If PetalLength is Medium and PetalWidth is Medium then class is versicolor
4) If PetalLength is Large and PetalWidth is Large then class is virginica

V. FINAL CONSIDERATIONS AND FUTURE WORK

Genetic fuzzy systems for classification have been widely researched and many proposals are available. Such methods take advantage of the global search of genetic algorithms and the interpretability of the produced models due to the fuzzy logic.

In this work, we detail the FCA-BASED RULE GENERATOR framework to automatically generate fuzzy classification systems using genetic algorithms and an approach to extract rules from data based on the formal concept analysis theory, namely, the FCA-EXTRACTION method.

FCA-EXTRACTION extracts rules directly from data in polynomial time, avoiding the extraction of rules with low
classification power. FCA-EXTRACTION extracts rules with different numbers of conditions in their antecedents, and does not require the definition of the number of rules to be extracted. FCA-EXTRACTION also allows the adjust of the number of extracted rules. Moreover, FCA-EXTRACTION can be used with any genetic approach that requires rule selection.

Although there are some free software for the generation of fuzzy systems available, as most of our proposal algorithms is new, the FCA-BASED RULE GENERATOR framework was implemented, using the Java language, and it is available for download. FCA-BASED RULE GENERATOR is distributed under the GNU GPL licence.

As future work, our main goal is to present the graphic interface of the FCA-BASED RULE GENERATOR framework. We also intend to include a module with previously proposed algorithms by the authors for feature selection and database definition.

ACKNOWLEDGMENTS

This research was partially financed by the Sao Paulo state research agency, FAPESP, and the Coordination for the Improvement of Higher Education Personnel, CAPES, Brazil.

REFERENCES