GF-Miner: a Genetic Fuzzy Classifier for Numerical Data

Vicky Tsikolidaki, Nikos Pelekis, Yannis Theodoridis

Dept of Informatics, Univ of Piraeus, 80 Karaoli-Dimitriou St, GR-18534 Piraeus, Greece

URL: http://infolab.cs.unipi.gr

E-mail: {vtsikol, npelekis, ytheod}@unipi.gr

Abstract Fuzzy logic and genetic algorithms are well-established computational techniques that have been employed to deal with the problem of classification as this is presented in the context of data mining. Based on *Fuzzy Miner* which is a recently proposed state-of-the-art fuzzy rule based system for numerical data, in this paper we propose *GF-Miner* which is a genetic fuzzy classifier that improves *Fuzzy Miner* firstly by adopting a clustering method for succeeding a more natural fuzzy partitioning of the input space, and secondly by optimizing the resulting fuzzy if-then rules with the use of genetic algorithms. More specifically, the membership functions of the fuzzy partitioning are extracted in an unsupervised way by using the fuzzy c- means clustering algorithm, while the extracted rules are optimized in terms of the volume of the rulebase and the size of each rule, using two appropriately designed genetic algorithms. The efficiency of our approach is demonstrated through an extensive experimental evaluation using the IRIS benchmark dataset.

1 Introduction

Computational Intelligence techniques such as fuzzy logic, artificial neural networks, and Genetic Algorithms (GA) are popular research domains, since they are able to confront intricate engineering problems. Gas are search algorithms, based on natural genetics that offer strong search capabilities in complex spaces. The basic idea is to preserve a population of chromosomes that evolves through a process of competition and controlled variation [1]. On the other hand, fuzzy rules are a collection of linguistic statements that describe how a fuzzy inference system should make a decision regarding classifying an input. They combine two or more input fuzzy sets and associate with them an output. Many researchers have used GA to optimize Fuzzy Rule-Based Systems (FRBS), which are known as Genetic Fuzzy Rule Based System (GFRBS), as in approaches [2, 3] where GA are used

for the automatic production of the knowledge base of a FRBS, which encodes the expert knowledge in the form of fuzzy rules.

In this paper we propose *GF-Miner* which is a genetic fuzzy classifier that is based on a fuzzy rule based system for numerical data, namely *Fuzzy Miner* [4]. *Fuzzy Miner* implements a heuristic fuzzy method for the classification of numerical data. *GF-Miner* adopts an improved unsupervised clustering method for succeeding a more natural fuzzy partitioning of the input space. Furthermore, a genetic process is devised that gets rid of needless rules from the initial set of rules and at the same time refines the rulebase by eliminating unnecessary terms from the fuzzy if-then rules. As such, *GF-Miner* optimizes not only the number of the produced rules but also their size constructing small rules of different sizes which are more comprehensible by the user and obtain higher classification accuracy.

Two are the most related works to the proposed approach. In [2] the authors create a rule base and optimize it by using two GA. The first GA constructs a rulebase of definite user-defined size. The number of fuzzy sets is static for every variable. The second GA reduces the fuzzy rule base by using the following fitness function: $FitnessFunction(C_i) = NPC(C_i) * (L - NR(C_i))$ where L is the number of rules generated in the previous stage, $NPC(C_i)$ is the number of Patterns Correctly Classified, and $NR(C_i)$ is the number of active rules.

In [3] the authors propose the construction of chromosome from already made fuzzy rules. The variables are separated also in a predefined number of fuzzy sets. The genetic algorithm codes the weights of attributes in the fuzzy rules. Every chromosome consists of real numbers that is the weights of the attributes in the fuzzy rules. The fitness function is the *Classification Accuracy Rate*.

In contradiction to the above approaches our work differs in three major points. First, *GF-Miner* allows the analyst to specify different number of fuzzy sets per input variable. Second, the number of generated rules is optimized by the GA and it is not predetermined by the user. Third, a more efficient fitness function is adopted, which as shown in the experiments, it obtains better results in terms of classification accuracy, while this is succeeded by maintaining a smaller rulebase.

Outlining the major issues that will be addressed in this paper, our main contributions are: a) *Fuzzy Miner* [4] is improved by proposing a new fuzzy partitioning method of the input space utilizing the Fuzzy C-Means (FCM) clustering algorithm [5], b) a genetic algorithm (i.e. GA_{SR}) is appropriately devised for the reduction of the size of the rules that are produced by the improved *Fuzzy Miner*, c) the number of the rules in the rulebase is reduced by the use of a second genetic algorithm (i.e. GA_{NR}) and d) the efficiency of our approach is demonstrated through an extensive experimental evaluation using the IRIS benchmark dataset.

The rest of the paper is structured as follows. Section 2 introduces the *GF-Miner*, describes its architectural aspects and the proposed fuzzy partitioning method. The proposed genetic process is described in Section 3, while Section 4 presents the results of our experimental study and evaluates our system. Finally, Section 5 provides the conclusions of the paper and some interesting research directions.

2 GF-Miner as an Extension of Fuzzy Miner

GF-Miner is based on *Fuzzy Miner* [4]. More specifically, it extends *Fuzzy Miner* by incorporating a more flexible and unsupervised way to partition the input space into fuzzy sets and improves it by optimizing its output with the use of GA.
Fuzzy Miner is composed of four principal components: a *fuzzification interface*, a *knowledge base*, a *decision-making logic* and a *defuzzification interface*. In *GF-Miner* we adapt the fuzzy partition in the database, and we optimize the rulebase of *Fuzzy Miner* (i.e. 1st level rulebase) in two different ways (i.e. 2nd and 3rd level rulebase). The architecture of *GF-Miner* is shown in Figure 1, while we elaborate on each of the components in the current and the following section.

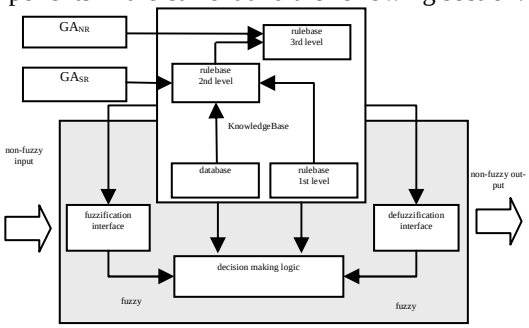

Fig. 1. GF-Miner architecture

The fuzzification interface performs a mapping that converts crisp values of input variables into fuzzy singletons. On the other end, the defuzzification interface performs a mapping from the fuzzy output of a FRBS to a crisp output.

Knowledge base: The knowledge base consists of a *database* and a *rulebase*.

Database - There are two factors that determine a database, i.e., a fuzzy partition of the input space and the membership functions of antecedent fuzzy sets. *GF-Miner* supports two types of membership functions, i.e. triangular and trapezoidal. The *Fuzzy Partition* partitions the input and output spaces to a sequence of fuzzy sets. In *GF-Miner* we use an unsupervised way to define the membership function by using the FCM clustering algorithm [5]. In Table 1 we show how we use the cluster centroids $V_i = \{V_1, V_2, \ldots V_{K_i}\}$, where K_i is the number of fuzzy sets for the *i*-th input variable x_i , to determine the parameters for every fuzzy set:

Table 1: Parameters for membership functions.

	Triangular	Trapezoidal
First fuzzy set:	$-\infty$, x_{imin} , V_2	$-\infty$, $-\infty$, x_{imin}, V_2
The next fuzzy sets: For j=1 to K_i -2:	V_j, V_{j+1}, V_{j+2}	V_j, $V_{j+1}-(V_{j+1}-V_j)/3$, $V_{j+1}+(V_{j+2}-V_{j+1})/3$, V_{j+2}
Last fuzzy set:	V_{K_i-1} , x_{imax}, $+\infty$	V_{K_i-1} , x_{imax}, $+\infty$, $+\infty$

Figure 2 depicts schematically the above fuzzy partition for a single variable.

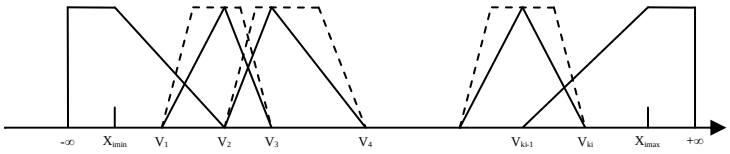

Fig. 2. Membership Function for Triangular and Trapezoidal

The rule base and the decision making logic are the same as in Fuzzy Miner [4].

3 The Genetic Process in GF-Miner

In this section we present two GA that we devise having as goal to optimize the rules that have been created so far. In detail the GA are used to reduce the number and the size of the rules. GA_{SR} reduces the size of the rules in the initial rulebase without eliminating any of them, while GA_{NR} reduces the number of the rules that have been produced after the GA_{SR} algorithm has been applied.

The GA_{SR} algorithm

Individual representation: The chromosome of an individual represents the antecedent part of the rule. The consequent part of the rule does not need to be coded and is the same as the consequent part of the corresponding rule.

Let *k* be the number of rules generated in the previous stage. Then a chromosome is composed of *k* genes, where each gene corresponds to a rule. Each *i* gene is partitioned into *n* binary fields where *n* is the number of input variables. We use 0 when the specific input variable is not important for the rule.

The GA_{SR} proceeds as follows:

Initial Population: It is generated randomly and we additionally introduce a chromosome that represents all rules, that is all genes will receive value 1.

Fitness Function: It is the number of the patterns correctly classified by the fuzzy rule base coded in the corresponding chromosome C_i: $FitnessFunction(C_i) = NPC(C_i)$, where $NPC(C_i)$ is the number of Patterns Correctly Classified.

Genetic Operators: For *selection* we use tournament selector and nonoverlapping populations. Furthermore, we use uniform *crossover* because this crossover operation does not take into account the position of every gene and the changes are randomly made. The *mutation* is done randomly according to the mutation probability and transforms 0 to 1 or 1 to 0. *As Stopping Condition* we used a maximum number of generations *m*. The new rule base is represented by the best chromosome with the best fitness value in the last generation.

Consequently, this GA reduces the number of rule conditions that are not important, thus their absence not affecting the number of Patterns Correctly Classified.

The GA_{NR} algorithm

As soon as the fuzzy rule base that has been created contains possibly redundant and/or unnecessary rules, and the aim of GA_{NR} is to eliminate some of them, having in mind that besides compactness the final rule base should continue giving high classification rates. In this genetic algorithm each individual encodes a set of prediction rules. The chromosomes are coded as a sequence of binary digits with the same length as the number of rules generated in the previous stage. Each gene is associated with one rule. If the binary digit is 1 the rule is active and the rule associated with this gene will be in the final rule base; otherwise will not. The crossover, mutation and stopping condition are the same we used in GA_{SR}. We use tournament selection and overlapping populations to reassure that the result will be the best, as in this case the best chromosomes of every generation are carried over to the next generation. As fitness function we use the following: $FitnessFunction(C_i) = CR(C_i)^2 * (L - NR(C_i) + 1)$, where $CR(C_i)$ is the Classification Rate, L is the number of rules generated in the previous stage and $NR(C_i)$ is the number of active rules. As a result we make sure that we keep a high Classification Rate and we decrease the number of active rules.

4 Evaluation of GF-MINER

We implemented the proposed method using C++ of Microsoft Visual Studio 6.0. The GA where implemented using the *GAlib* which is an object-oriented library of GA components [6]. The aim of our evaluation is twofold: on the one hand, we compare the classification accuracy of *GF-Miner* with the one of the initial FRBS *Fuzzy Miner* [4], while on the other hand, we compare it with two state-of-the-art genetically optimized approaches, namely [2] and [3], which are the most related to our approach. We used the IRIS dataset obtained from UCI repository of machine learning databases [7], which consists of 50 samples from each of 3 species of Iris flowers (*setosa*, *virginica* and *versicolor*) and 4 features were measured from each sample: the length and the width of sepal and petal.

The data set was partitioned randomly into training and test subset in two ways: *Partition A (70%)*: 105 instances for training and 45 instances for test.

Partition B (50%): 75 instances for training and 75 instances for test.

We used three fuzzy sets for every variable with triangular membership function to be compared with [3] and [2] where the writers use both triangular membership functions with three fuzzy sets. The classification rate and the number of rules are calculated as the average after 10 runs. We also show the minimum and the maximum values achieved by each partition.

The experimental results shown in table 2 prove that our approach presents better results than the other approaches as we can achieve a higher classification rate and still generate few rules. We observe that Fuzzy Miner has high classification rate but constructs many rules. As an improvement we see that GF-Miner in-

creases the classification rate and at the same time reduces dramatically the number of the rules.

Table 2. Experimental Results

Partition	Approach	Classification Rate	#Rules
A	GF-Miner	Avg: 97,56 Max: 97,78Min: 95,56	Avg: 4,9 Max: 7 Min: 3
	Fuzzy Miner	95,56	15
	[2]	96,9	4
B	GF-Miner	Avg: 97,87 Max: 98,67 Min: 97,33	Avg: 4,6 Max: 6 Min: 4
	Fuzzy Miner	93,33	14
	[3]	96,33	3

5 Conclusions and Future Work

In this paper we propose *GF-Miner* which is a genetic fuzzy classifier that improves *Fuzzy Miner* [4] which is a recently proposed state-of-the-art FRBS for numerical data. More specifically, we used the FCM clustering algorithm to succeed a more natural definition of the membership functions of the fuzzy partition, while the extracted rules are optimized as far as the volume of the rulebase and the size of each rule is concerned, using two appropriately designed genetic algorithms. As future work we plan to evaluate *GF-Miner* using high dimensional datasets. Another direction will be to further improve the genetic algorithms to minimize their computational cost.

References

1. Cordón O, Gomide F, Herrera F, Hoffmann F, Magdalena L (2004) Ten years of genetic fuzzy systems: Current framework and new trends, Fuzzy Sets and Systems 41:5-31
2. Castro P, Camargo H (2005) Improving the genetic optimization of fuzzy rule base by imposing a constraint condition on the number of rules, V Artificial Intelligence National Meeting (ENIA), São Leopoldo, Rio Grande de Sul 972-981
3. Chen SM, Lin HL (2006) Generating weighted fuzzy rules from training instances using genetic algorithms to handle the Iris data classification problem. Journal of Information Science and Engineering 22: 175-188
4. Pelekis N, Theodoulidis B, Kopanakis I, Theodoridis Y (2005) Fuzzy Miner: Extracting Fuzzy Rules from Numerical Patterns. International Journal of Data Warehousing and Mining 57-81
5. Bezdek JC (1981) Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York
6. Wall M (1996) GAlib: A C++ Library of Genetic Algorithm Components, version 2.4, Documentation Revision B, Massachusetts Institute of Technology. http://lancet.mit.edu/ga/. Accessed January 19 2009
7. Blake CL, Merz CJ, (1998) UCI Repository of machine learning databases, Irvine, University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html. Accessed January 19 2009