Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution

Roberto R. F. Mendes   Fabricio de B. Voznika   Alex A. Freitas   Julio C. Nievola

PUC-PR
PPGIA - CCET
Av. Imaculada Conceição, 1155
Curitiba - PR, 80215-901 Brazil
{alex, nievola}@ppgia.pucpr.br
http://www.ppgia.pucpr.br/~alex
+55 41 330-1669

Abstract. In essence, data mining consists of extracting knowledge from data. This paper proposes a co-evolutionary system for discovering fuzzy classification rules. The system uses two evolutionary algorithms: a genetic programming (GP) algorithm evolving a population of fuzzy rule sets and a simple evolutionary algorithm evolving a population of membership function definitions. The two populations co-evolve, so that the final result of the co-evolutionary process is a fuzzy rule set and a set of membership function definitions which are well adapted to each other. In addition, our system also has some innovative ideas with respect to the encoding of GP individuals representing rule sets. The basic idea is that our individual encoding scheme incorporates several syntactical restrictions that facilitate the handling of rule sets in disjunctive normal form. We have also adapted GP operators to better work with the proposed individual encoding scheme.

1 Introduction

In the context of machine learning and data mining, one popular way of expressing knowledge consists of IF-THEN rules. This is due to the fact that they are intuitively comprehensible to a human being [5]. In addition, they represent independent units of knowledge, so that alterations can easily take place in their contents. IF-THEN rules are composed of two parts. The first part (IF component, or rule antecedent) corresponds to a conjunction of conditions that, if verified true, imply that the condition contained in the second part (THEN component, or rule consequent) is also considered true.

Rules in their classic format are appropriate when their conditions are constituted by discrete or categorical variables. However, the presence of continuous variables creates situations that thwart the common sense. Let's consider the rule: "IF age < 25 THEN safe_driver = no". The problem here is the sudden and unnatural transition between categories: an individual can be classified as not being a safe driver today but, in the following day, he might have completed 25 years and thus be classified as being a safe driver. This could lead a data mining system to completely different predictions in the interval of a single day. One promising alternative to work with continuous variables and to overcome this inconvenience is the use of fuzzy logic. Besides expressing knowledge in a more natural way, fuzzy logic is also a flexible and powerful method for uncertainty management [13], [6].

In the literature several techniques have been used for discovery of fuzzy IF-THEN rules. Several recent projects have proposed the use of evolutionary algorithms for fuzzy rule discovery [2], [10], [11], [19], [21], [17], because it allows a global search in the state space, increasing the probability of converging to the globally-optimal solution.

The main characteristic of our proposed system that makes it different from the above systems it that our system is based on the co-evolution of fuzzy rule sets and membership function definitions, using two separate populations, whereas in general the above projects are based on the evolution of a single population. The population of fuzzy rule sets is evolved by a Genetic Programming (GP) algorithm, whereas the population of membership function definitions is evolved by a simple evolutionary algorithm.

In addition, our system also has some innovative ideas with respect to the encoding of GP individuals representing rule sets. The basic idea is that our individual encoding scheme incorporates several syntactical restrictions that facilitate the handling of rule sets in disjunctive normal form. We have also adapted GP operators to better work with the proposed individual encoding scheme.

The remainder of this paper is organized as follows. Section 2 describes in detail our proposed co-evolutionary system. Section 3 discusses computational results. Finally, section 4 concludes the paper.

2 The Proposed Co-Evolutionary System for Fuzzy Rule Discovery

2.1 An Overview of the System

This section presents an overview of our CEFR-MINER (Co-Evolutionary Fuzzy Rule Miner) system. CEFR-MINER is a system developed for the classification task of data mining. It consists of two co-evolving evolutionary algorithms. The first one is a Genetic Programming (GP) algorithm where each individual represents a fuzzy rule set. A GP individual specifies only the attribute-value pairs composing the rule conditions of that individual's rule set. The definitions of the membership functions necessary to interpret the fuzzy rule conditions of an individual are provided by the second population. The second algorithm is a simple evolutionary algorithm, which works with a "population" of a single individual. This population evolves via the principle of natural selection and application of mutation, but not crossover. This single individual specifies definitions of all the membership functions for all attributes being fuzzified (all originally continuous attributes). These definitions are used by the first population of GP individuals, as mentioned above. Note that categorical attributes are not fuzzified – their values are handled only by the GP population.

As a result, the system simultaneously evolves both fuzzy rule sets and membership function definitions specifically suited for the fuzzy rule sets. The main advantage of this co-evolutionary approach is that the fitness of a given set of membership function definitions is evaluated across several fuzzy rule sets, encoded into several different GP individuals, rather than on a single fuzzy set. This improves the robustness of that evaluation.

This basic idea of co-evolution for fuzzy-rule discovery has been recently proposed by [4]. The main differences between this work and our system are as follows. (a) Delgado et al.'s work uses three co-evolving populations and our work uses only two; (b) Delgado et al.'s work uses genetic algorithms for evolving two of its three populations. By contrast, we use genetic programming to evolve the rule set population; (c) our work addresses the classification task of data mining, whereas Delgado et al.'s work addresses the problem of numeric function approximation.

2.2 The Genetic Programming Population

2.2.1 Rule Representation

Our system follows the Pittsburgh approach [7] and thus each individual represents a set of rules. Each rule has the form: IF conditions THEN prediction. The prediction of the rule has the form: "goal attribute = class", where class is one of the values that can be taken on by the goal attribute. In each run of the system all individuals of the GP population are associated with the same prediction. Therefore, there is no need to explicitly encode this prediction into the genome of an individual. Since each run discovers rules predicting a single class, the system must be run *c* times, where *c* is the number of classes. Although this approach increases processing time, it has two important advantages: (a) it simplifies individual encoding; and (b) it avoids the problem of mating between individuals that predict different classes, which could produce low-quality offspring. Each individual actually corresponds to a set of rule antecedents encoded in disjunctive normal form (DNF), such as: (sore throat = true AND age = *low*) OR (headache = true AND NOT temperature = *low*).

In our system the function set contains the logical operators {AND, OR, NOT}. Since each individual represents fuzzy rules, a fuzzy version of these logical operators must be used. We have used the standard fuzzy AND (intersection), OR (union) and NOT (complement) operators [13]. More precisely, let $\mu_A(x)$ denote the membership degree of an element x in the fuzzy set A, i.e. the degree to which x belongs to the fuzzy set A. The standard AND of two fuzzy sets A and B, denoted A AND B, is defined as $\mu_{A\text{-AND-}B}(x) = \min[\mu_A(x),\mu_B(x)]$, where min denotes the minimum operator. The standard OR of two fuzzy sets A and B, denoted A OR B, is defined as $\mu_{A\text{-OR-}B}(x) = \max[\mu_A(x),\mu_B(x)]$, where max denotes the maximum operator. The standard NOT of a fuzzy set A, denoted NOT A, is defined as $\mu_{\text{NOT-}A}(x) = 1 - \mu_A(x)$.

The terminal set consists of all possible conditions of the form: "$\text{Attr}_i = \text{Val}_{ij}$", where Attr_i is the i-th attribute of the dataset. If attribute Attr_i is categorical, Val_{ij} is the j-th value of the domain of $Attr_i$. If attribute $Attr_i$ is continuous - which means it is being fuzzified by the system - Val_{ij} is a linguistic value in {*low*, *medium*, *high*}. We use only three linguistic values in order to reduce the size of the search space.

In order to produce individual trees with only valid rule antecedents and in DNF we propose some syntactic restrictions in the tree representation, as follows: (a) the root node is always an OR node; (b) with the exception of the root node, each OR node must have as its parent another OR node, and can have as its children any kind of node; (c) each AND node must have as its parent either an OR node or another AND node, and can have as its children AND, NOT or terminal nodes; (d) a NOT node can have as its parent an OR node, an AND node or a NOT node; and it can have as its child either another NOT node or a terminal node (we allow conditions of the form "NOT NOT ..." to allow the possibility of a NOT being cancelled by another NOT as a result of genetic operators) and (e) there cannot be two or more terminal nodes referring to the same attribute in the same rule antecedent, since this would tend to produce invalid rule antecedents such as (sex = male AND sex = female). Fig. 1 shows an individual with five rule antecedents.

These syntactic constraints are enforced both when creating individuals of the initial population and when modifying individuals due to the action of a genetic operator. This approach can be regarded as a kind of strongly-typed GP [16] proposed specifically for the discovery of rule sets in disjunctive normal form, which makes it attractive for data mining applications.

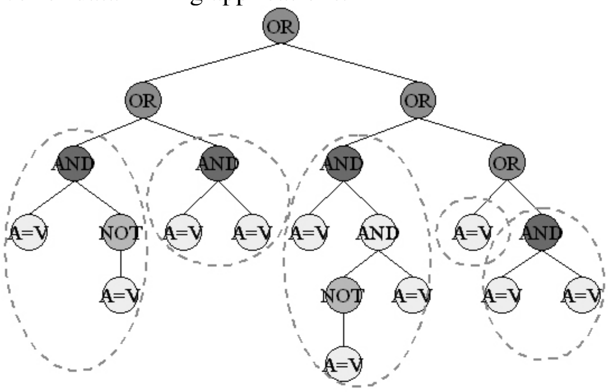

Fig. 1. A tree representing five rule antecedents

The main advantage of working with the DNF directly into the tree representation, rather than converting a rule set into DNF after GP has evolved, is that this makes it easier to fulfil the aforementioned restriction (e). Because of the hierarchical position of the nodes, it is easy to collect the terminal nodes of an individual rule, as shown in Fig. 1, in order to check whether or not a condition can be inserted into that rule.

Another possible approach to assure that the GP will run only with syntactically valid individuals would be to use a context-free grammar to implement the aforementioned syntactic restrictions. The drawback of this approach would be the difficulty in checking syntactic restriction (e), which would lead to an explosion of the number of production rules in the grammar. To avoid this, a logic grammar could be used [20], but this would introduce some complexity to the system. Thus, we have preferred the above-described direct implementation of syntactic constraints.

2.2.2 Selection and Genetic Operators

We use the tournament selection method, with tournament size 2 and with a simple extension: if two individuals have the same fitness, the one with smaller complexity is selected. Complexity is measured by the following formula [12]:

$$\text{complexity} = 2 \times \text{number\_of\_rules} + \text{number\_of\_conditions} \,. \qquad \textbf{(1)}$$

This extension was motivated by observations in our experiments: sometimes the two individuals competing in the tournament had the same fitness value, even though they were different individuals.

Once two individuals are selected crossover is performed in a similar way to conventional GP crossover, with the difference that in our case the crossover operator respects the above-discussed syntactic restrictions, in order to guarantee that crossover always generates syntactically-valid offspring. (If crossover cannot produce syntactically-valid individuals, the crossover operation fails and no children are produced.)

The current version of the system uses a crossover probability of 80%, a relatively common setting in the literature. However, in our system the offspring produced by crossover is not necessarily inserted into the population. Our population updating strategy is as follows. Once all crossovers have been performed, all the produced offspring are added to the population of individuals. Therefore, the population size is provisionally increased by 80%. Then all individuals are sorted by fitness value, and the worst individuals are removed from the population. The number of removed individuals is chosen in such a way that the number of individuals left in the population is always a constant population size, set to 250 individuals (an empirically-determined setting) in our experiments. We chose this population-updating strategy mainly because it increases selection pressure, in comparison with a conventional generational-replacement strategy. This is analogous to the (μ+λ)-strategy employed in the second EA of this system, described in Section 2.3.2. The main difference is that here we use a (μ+λ) strategy on top of tournament selection, whereas the classic (μ+λ) strategy uses no such scheme.

Our system uses a mutation operator where a node is randomly chosen and then the subtree rooted at that node is replaced by another randomly-generated subtree. In the current version of the system an individual undergoes mutation with a probability of 20% (an empirically-determined setting), with just one exception. The best individual of each generation never undergoes mutation, so that its fitness will never be worsened.

2.2.3 Fitness Function

In order to calculate the fitness of a GP individual, the first step is to compute the following counters:

- *TP* (true positives) is the number of examples that are covered by at least one of the individual's rules and have the class predicted by those rules;
- *FP* (false positives) is the number of examples that are covered by at least one of the individual's rules but have a class different from the class predicted by those rules;
- *FN* (false negatives) is the number of examples that are not covered by any of the individual's rules but have the class predicted by those rules;
- *TN* (true negatives) is the number of examples that are not covered by any of the individual's rules and do not have the class predicted by those rules.

Note that the true positives and true negatives correspond to correct predictions made by the individual being evaluated, whereas the false positives and the false negatives correspond to wrong predictions made by that individual. In our system the fitness of a GP individual is computed by the following formula [9]:

$$(TP / (TP + FN)) \times (TN / (FP + TN)) . \qquad \textbf{(2)}$$

In the data mining literature, in general it is implicitly assumed that the values of TP, FP, FN and TN are crisp. This very commonplace assumption is invalid in our case, since our system discovers fuzzy rules. In our system an example can be covered by a rule antecedent to a certain degree in the range [0..1], which corresponds to the membership degree of that example in that rule antecedent. Therefore, the system computes fuzzy values for TP, FP, FN and TN.

The membership degree of record *r* into the rule set encoded by the individual *I* is computed as follows. For each rule of *I*, the system computes the membership degree of *r* into each of the conditions of that rule. Then the membership degree for the entire rule antecedent is computed by a fuzzy AND of the membership degrees for all the rule conditions. This process is repeated for all the rules of the individual *I*. Then the membership degree of the entire rule set is computed by a fuzzy OR of the membership degrees for all the rules of *I*.

For instance, suppose a training example has the class predicted by the individual *I*'s rules. Ideally, we would like that example to be covered by at least one of *I*'s rules to a degree of 1, so that the entire rule set of *I* would cover that example to a degree of 1. Suppose that *I* has two rules, and that the current training example is covered by those rules to degrees of 0.6 and 0.8. Then the fuzzy OR would return a membership degree of 0.8 for the entire rule set. This means that the prediction made by the individual is 80% correct and 20% wrong. As a result, this example contributes a value of 0.8 for the number of true positives and a value of 0.2 for the number of false negatives.

2.2.4 Tree Pruning

Rule pruning is important not only in data mining [3] but also in GP, due to the well-known effects of code bloat [14], [1]. Code bloat has greatly affected our system's performance. In our initial experiments, with no pruning at all, some datasets required an unacceptable amount of running time. Therefore, we have designed an operator to prune GP trees. The basic idea of this operator is to randomly remove conditions from a rule with a null coverage – i.e. a rule which does not cover any record – until it covers at least one record or until all conditions are removed, which corresponds to removing the entire rule from its rule set.

More precisely, each rule of the individual is separated and evaluated by itself. The ones that have a null coverage will have some conditions dropped according to the following criteria:

- If a rule has more than 7 conditions, some conditions are randomly removed until the rule has between 5 and 7 conditions (a randomly chosen number). If even after this step the rule remains with a null coverage, the next criterion will be applied;
- If the number of conditions of a rule is less than or equal to 7, its conditions will be dropped randomly one by one until the rule covers at least one record or all of its conditions are dropped, removing the rule completely from the individual.

This operator is applied to an individual with a 20% probability. However, as an individual might be worsened by this operator, it is never applied to the best individual of the current generation. The motivation to apply the above operator only to 20% of the individuals is to save processing time, since this is a relatively computationally-expensive operator.

After the end of the evolution, the best individual also undergoes a different tree pruning. This operator removes two kinds of redundant rules: rules with a null coverage and duplicate rules. This final tree pruning does not alter the fitness of the individual, since the removal of null-coverage/duplicate rules does not alter the set of examples covered by an individual's rule set.

2.3 The "Population" of Membership Functions

As mentioned above, in our system the values of all continuous attributes are fuzzified into three linguistic values, namely low, medium, and high. These linguistics values are defined by trapezoidal membership functions. Each continuous attribute is associated with its own membership functions. Hence, the membership functions are dynamically evolved, modifying a set of parameters defining the membership functions, to get better adapted to their corresponding attribute. All the parameters of all membership functions are encoded into a single individual. This individual is considered as a "population" (in a loose sense of the term, of course) separated from the GP population. As mentioned in section 2.1, this single-individual population co-evolves with the GP population.

2.3.1 Individual Representation

The individual is divided into *k* parts (or "chromosomes", loosely speaking), where *k* is the number of attributes being fuzzified. Each chromosome consists of four genes, denoted *g1*, *g2*, *g3* and *g4*, which collectively define the three membership functions (low, medium, and high) for the corresponding attribute, as shown in Fig. 2. Each gene represents an attribute value that is used to specify the coordinate of two trapezoid vertices belonging to a pair of "adjacent" membership functions. The system ensures that $g1 \leq g2 \leq g3 \leq g4$.

This individual representation has two advantages. First, it reduces the search space of the evolutionary algorithm and saves processing time, since the number of parameters to be optimized by the evolutionary algorithm is reduced. Second, this representation enforces some overlapping between "adjacent" membership functions and guarantees that, for each original value of the continuous attribute, the sum of its degrees of membership into the three linguistic values will be 1, which is intuitively sensible.

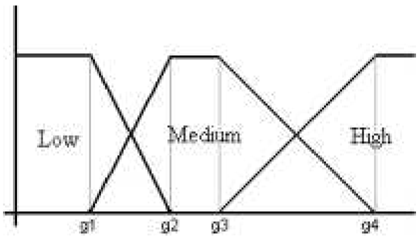

Fig. 2. Definition of 3 trapezoidal membership functions by 4 genes (g1, g2, g3, g4)

2.3.2 Evolutionary Algorithm to Evolve Membership Functions

Obviously, it is not possible to perform crossover in the single-individual "population" of membership functions. Therefore, the evolution of the single individual representing membership functions is the result of a simple evolutionary algorithm, which evolves by means of a (μ+λ)-evolution strategy (more specifically the (1+5)-strategy), described as follows.

First of all, the individual is cloned 5 times. Each clone is an exact copy of the original individual. Then the system applies to each clone a relatively high rate of mutation. Each chromosome (i.e. a block of four contiguous genes, g1, g2, g3 and g4, defining the membership functions of a single attribute) has an 80% probability of undergoing a single-gene mutation. The mutation in question consists of adding or subtracting a small randomly-generated value to the current gene value. This has the effect of shifting the coordinate of the trapezoid vertices associated with that gene a little to the right or to the left.

Note that, since a chromosome has four genes and only one of those genes is mutated, a mutation rate of 80% per chromosome corresponds to a mutation rate of 20% per gene. Our motivation to use this relatively high mutation rate is the desire to perform a more global search in the space of candidate membership function definitions. If we used a much smaller mutation rate, say 1% or 0.1%, probably at most one gene of an entire individual (corresponding to all attributes being fuzzified) would be modified. This would correspond to a kind of local search, where a new candidate solution being evaluated (via fitness function) would differ from its "parent" solution by only one gene, without taking into account gene interactions. By contrast, in our (1+5)-evolution strategy scheme a new candidate solution being evaluated differs from its "parent" solution by several genes, and the effect of all these gene modifications is evaluated as a whole, taking into account gene interactions. This is important, since the attributes being fuzzified can interact in such a way that modifications in their membership functions should be evaluated as a whole. Actually, the ability to take into account attribute interactions can be considered one of the main motivations for using an evolutionary algorithm, rather than a local search algorithm.

In any case, once the 5 clones have undergone mutation, the 5 just-generated individuals are evaluated according to a fitness function – which is discussed in the next subsection. The best individual is kept and all others are discarded.

The number of clones (5) used in our experiments was empirically determined as a good trade-off between membership-function quality and processing time.

2.3.3 Fitness Function

Recall that the individual of the membership-function population represents definitions of membership functions to be used for defining rule antecedents being evolved by the GP population. Hence, the quality of the individual of the former population depends on the predictive accuracy of individuals of the latter population. More precisely, in our co-evolutionary scheme the fitness value of the membership-function individual is computed as the sum of the fitness values of a group of individuals of the GP population. To compute the fitness, the system uses only a small portion of the GP population – for the experiments reported in this paper we used the best five individuals – , in order to reduce processing time.

2.4 Classifying New Examples

Recall that a complete execution of our system generates one rule set for each class found in the data set. These rule sets are then used to classify the examples of the test set. For each test example the system computes the degree of membership of that example to each rule set (each one predicting a different class). Then the example is assigned the class of the rule set in which the example has the largest degree of membership. The accuracy rate on the test set is computed as the number of correctly classified test examples divided by the total number of test examples, as usual in the classification literature.

3 Computational Results

We have evaluated our system across four public-domain data sets from the UCI (University of California at Irvine) data set repository. These data sets are available from http://www.ics.uci.edu/~mlearn/MLRepository.html. Some of these data sets had a small number of records with unknown values. Since the current version of our system cannot cope with this problem, those records were removed. All the results reported below were produced by using a 10-fold cross-validation procedure [9].

In order to evaluate the performance of our system we have compared it to two other evolutionary systems found in the literature: ESIA [15] and BGP [18]. Both ESIA and BGP discover crisp rules. They were chosen for comparison because they have been applied to some of the data sets used in our experiments and because they have obtained good results in comparison with other data mining systems. The results for ESIA and BGP reported here are taken directly from the above-mentioned papers. The results for ESIA were also produced by 10-fold cross-validation, whereas the results for BGP were produced by generating 30 training and test sets.

As can be seen in Table 1, our system and ESIA obtained the same accuracy rate on the Iris data set. (The numbers between brackets for our system are standard deviations.) On the other two data sets (CRX and Heart), our system considerably outperforms ESIA. Our system outperforms BGP on the Iris data set, but BGP outperforms our system on the Ionosphere data set.

A possible explanation for the lower performance of the fuzzy rules discovered by our system in the Ionosphere data set is suggested by the large number (34) of continuous attributes in that data set. This suggests the possibility that the simple evolutionary algorithm described in section 2.3 has difficulty in coping with such a relatively high number of attributes being fuzzified. In other words, in this case the size of the search space may be too large for such a simple evolutionary algorithm. This hypothesis will be further investigated in future work.

Table 1. Accuracy rate (on test set), in %, of our system , ESIA and BGP.

Data set	Our system	ESIA	BGP
CRX	84.7 (±3.5)	77.39	N/A
Heart (statlog)	82.2 (±7.1)	74.44	N/A
Ionosphere	88.6 (±6.0)	N/A	89.2
Iris	95.3 (±7.1)	95.33	94.1

Overall, we consider these results very promising, bearing in mind that, unlike ESIA and BGP, our system has the advantage of discovering fuzzy rules, which tend to be more intuitive for a user than the "hard" thresholds associated with continuous attributes in crisp rules. On the other hand, like most evolutionary algorithms, our co-evolutionary system needs a good amount of computational time to run. More precisely, a single iteration of cross-validation took a processing time varying from a couple of minutes for the Iris data set to about one hour for the CRX data set – results obtained for a dual-processor Pentium II 350. Shorter processing times may be obtained by the use of parallel data mining techniques [8], but this point is left for future research.

4 Conclusions and Future Research

We have proposed a co-evolutionary system for discovering fuzzy classification rules. The system uses two evolutionary algorithms: a genetic programming (GP) algorithm evolving a population of fuzzy rule sets and a simple evolutionary algorithm evolving a population of membership function definitions. The two populations co-evolve, so that the final result of the co-evolutionary process is a fuzzy rule set and a set of membership function definitions that are well adapted to each other.

The main advantage of this co-evolutionary approach is that the fitness of a given set of membership function definitions is evaluated across several fuzzy rule sets, encoded into several different GP individuals, rather than on a single fuzzy set. This makes that evaluation more robust. In order to mitigate the problem of long processing times, our system evaluates a set of membership function definitions only across the few best GP individuals.

In addition, our system also has some innovative ideas with respect to the encoding of GP individuals representing rule sets. The basic idea is that our individual encoding scheme incorporates several syntactical restrictions that facilitate the handling of rule sets in disjunctive normal form. We have also adapted GP operators to better work with the proposed individual encoding scheme.

We have evaluated our system across four public domain data sets and compared it with two other evolutionary systems (ESIA and BGP) found in the literature which used the same data sets. Our results can be summarized as follows:

(a) Our co-evolutionary system considerably outperforms ESIA in two out of three datasets and equals it in the other data set, with respect to predictive accuracy.

(b) Our system is competitive with BGP in two data sets. (In one data set our system outperforms BGP, whereas BGP outperforms our system in the other data set.)

(c) Our system has the advantage of discovering fuzzy rules, which tend to be more intuitive for the user than the crisp rules discovered by ESIA and BGP.

There are several directions for future research. For instance, the GP tree pruning operator currently used in our system is a “blind” operator, in the sense that tree nodes to be pruned are randomly chosen. It seems that a promising research direction would be to design a more “intelligent” pruning operator, which would choose the tree nodes to be pruned based on some estimate of the predictive power of those tree nodes.

Note that the above suggested research direction concerns improvement in the GP algorithm used by our system. However it seems that the most important point to investigate in future research is the performance of the simple evolutionary algorithm for evolving membership function definitions. It is possible that the current version of this algorithm is not robust enough to cope with data sets having a large number of attributes being fuzzified. This hypothesis must be further investigated in the future, which might lead to improvements in the current version of this simple evolutionary algorithm.

References

1. W. Banzhaf, P. Nordin, R.E. Keller, Francone FD Genetic Programming ~ an Introduction. Morgan Kaufmann, 1998.

2. P.J. Bentley. "Evolutionary, my dear Watson" - investigating committee-based evolution of fuzzy rules for the detection of suspicious insurance claims. Proc. Genetic and Evolutionary Computation Conf. (GECCO-2000), 702-709. Morgan Kaufmann, 2000.
3. L.A. Breslow and D.W. Aha. Simplifying decision trees: a survey. The Knowledge Engineering Review, 12(1), 1-40. Mar. 1997.
4. M. Delgado, F.V. Zuben and F. Gomide. Modular and hierarchical evolutionary design of fuzzy systems. Proc. Genetic and Evolutionary Computation Conf. (GECCO-99), 180-187. Morgan Kaufmann, 1999.
5. U.M. Fayyad, G. Piatetsky-Shapiro and P. Smyth. From data mining to knowledge discovery: an overview. In: U.M. Fayyad et al. (Eds.) Advances in Knowledge Discovery & Data Mining, 1-34. AAAI/MIT, 1996.
6. C.S. Fertig, A.A. Freitas, L.V.R. Arruda and C. Kaestner. A Fuzzy Beam-Search Rule Induction Algorithm. Principles of Data Mining and Knowledge Discovery (Proc. 3rd European Conf. - PKDD-99). Lecture Notes in Artificial Intelligence 1704, 341-347. Springer-Verlag, 1999.
7. A.A. Freitas. A survey of evolutionary algorithms for data mining and knowledge discovery. To appear in: A. Ghosh and S. Tsutsui. (Eds.) Advances in Evolutionary Computation. Springer-Verlag, 2001.
8. A.A. Freitas and S.H. Lavington. Mining Very Large Databases with Parallel Processing. Kluwer Academic Publishers, 1998.
9. D.J. Hand. Construction and Assessment of Classification Rules. John Wiley&Sons, 1997.
10. H. Ishibuchi and T. Nakashima. Linguistic rule extraction by genetics-based machine learning. Proc. Genetic and Evolutionary Computation Conf. (GECCO-2000), 195-202. Morgan Kaufmann, 2000.
11. H. Ishibuchi, T. Nakashima and T. Kuroda. A hybrid fuzzy GBML algorithm for designing compact fuzzy rule-based classification systems. Proc. 9th IEEE Int. Conf. Fuzzy Systems (FUZZ IEEE 2000), 706-711. San Antonio, TX, USA. May 2000.
12. C.Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. Machine Learning 13, 189-228. 1993.
13. G.J. Klir and B. Yuan. Fuzzy Sets and Fuzzy Logic. Prentice-Hall, 1995.
14. W.B. Langdon, T. Soule, R. Poli and J.A. Foster. The evolution of size and shape. In: L. Spector, W.B. Langdon, U-M. O'Reilly and P.J. Angeline. (Eds.) Advances in Genetic Programming Volume 3, 163-190. MIT Press, 1999.
15. J.J. Liu and J.T. Kwok. An Extended Genetic Rule Induction Algorithm. Proc. Congress on Evolutionary Computation (CEC-2000). La Jolla, CA, USA. July 2000.
16. D.J. Montana. Strongly typed genetic programming. Evolutionary Computation 3(2), 199-230. 1995.
17. C.A. Pena-Reyes and M. Sipper. Designing breast cancer diagnostic systems via a hybrid fuzzy-genetic methodology. Proc. 8th IEEE Int. Conf. Fuzzy Systems. 1999.
18. S.E. Rouwhorst and A.P.Engelbrecht. Searching the Forest: Using Decision Tree as Building Blocks for Evolutionary Search in Classification. Proc. Congress on Evolutionary Computation (CEC-2000), 633-638. La Jolla, CA, USA. July 2000.
19. D. Walter and C.K. Mohan. ClaDia: a fuzzy classifier system for disease diagnosis. Proc. Congress on Evolutionary Computation (CEC-2000), 1429-1435. La Jolla, CA. 2000.
20. M.L. Wong and K.S. Leung. Data Mining Using Grammar Based Genetic Programming and Applications. Kluwer, 2000.
21. N. Xiong and L. Litz. Generating linguistic fuzzy rules for pattern classification with genetic algorithms. Principles of Data Mining and Knowledge Discovery (Proc. PKDD-99) Lecture Notes in Artificial Intelligence 1704, 574-579. Springer-Verlag, 1999.