AREX – Classification Rules Extracting Algorithm Based on Automatic Programming

Vili Podgorelec[1] and **Peter Kokol**[1] and **Ivan Rozman**[1]

Abstract. The paper presents a hybrid classification method of BNF grammar-based genetic programming and evolutionary decision tree induction, customized for the rule induction according to a layered hierarchical scheme – the AREX approach. It incorporates two original, independent evolutionary algorithms which together solve the problem of automatic classification rules induction. The method is applied to five real world databases (from medicine and software engineering) and the results are compared to those obtained with C5/See5 to evaluate the method's efficiency. Ideally, this paper will inspire future research in this same area and along similar lines.

1 INTRODUCTION

In the past half a century artificial intelligence research has not fulfilled many expectations and has left a lot of challenges open. One of the very actual between them is to find a general machine learning and decision making technique that works on real world data and simultaneously include specific limitations of several application areas but still not limit its all-purposeness.

It has been tried to solve the challenge with so called new artificial intelligence, which most evident members are intelligent systems. They are based on a variety of methodologies, one of the most interesting of them is automatic programming. The goal of automatic programming is to automatically construct a computer program for solving a given problem. Automatic programming is domain independent approach to problem solving and as such appropriate basic technique for machine learning in intelligent systems.

Our work concentrates on design and verification of an intelligent system for automatic knowledge discovery with automatic programming. For this purpose we: 1) analyzed performance and adequacy of the existing automatic knowledge discovery systems, 2) identified disadvantages of the existing intelligent systems, 3) defined and verified general model for intelligent systems for automatic knowledge discovery based on automatic programming, and 4) analyzed performance of the developed intelligent system with the emphasis on the real world problems from medicine and software evaluation.

2 THE AREX ALGORITHM

Knowledge discovered with data mining algorithms should ideally give an in-depth explanation of a problem domain along with a good classification [5]. Experts are performing exhaustive analyses of the results, which are the output of knowledge discovery tools, in order to extract the useful knowledge. To make a part of their work as easy as possible the best way is to present the results in a form of a set of classification rules, which are clear and straightforward to understand, accept or reject. Ideal system would therefore include:

- accuracy – classification with minimal error rate,
- compactness – use of a minimum number of rules, and
- simplicity – single rules are not complex.

Data to use as the source for knowledge discovery system are represented with the set of training objects $o_1, \ldots, o_N$. Each training object o_i is described with the values of attributes $a_{i1}, \ldots, a_{ik}$ and the accompanied decision ω_i from the set of m possible decisions $[\Omega_1, \ldots, \Omega_m]$. Attributes can be either numeric (value is a numbers from a continues interval) or discrete (value is one from the discrete set of all possible values). It is not necessary for all values to be known, as the algorithm can work also with missing values.

Knowledge that is discovered with the help of our algorithm is represented with a set of classification rules. Each single rule in a set is in the following form:

`if <condition> then <decision>`, where

`<condition> := <c1> and … and <cd>`, and
`<decision>` := ω, $\omega \in [\Omega_1, \ldots, \Omega_2]$

In this manner the rules are clear and easy enough for further analyses and in the same time their functional power is strong enough for successful classification. A set of classification rules can lose on the understandability if the number of rules in a set is too high. On the other hand the classification accuracy would decrease if the number of classification rules in a set is too low. Following this statement all the rules are arranged regarding their importance into several levels to form the hierarchical classification model (figure 1).

An approach to the distribution of data objects into subsets that will be covered by the rules on a specific level is defined first. For this purpose an algorithm for the construction of decision trees was defined [8] – with the help of classification by decision trees it can be determined which objects are appropriate to be used on a specific level of classification.

The complete rules extracting algorithm has been called AREX (*Automatic Rules EXtractor*). AREX uses as the input a training set of objects and based on those objects classification rules are built, hierarchically distributed over several levels. Algorithm AREX includes a hybrid system of two basic algorithms: 1) an evolutionary algorithm for the construction of decision trees that is used to distribute the objects between levels and to build a part of the initial set of classification rules, and 2) *proGenesys* system that allows automatic evolution of programs in an arbitrary programming language and is used for the construction of classification rules on a single level. The basic outline of the AREX algorithm is illustrated in figure 2 and can be described in the following steps:

[1] University of Maribor – FERI, Smetanova 17, SI-2000 Maribor, Slovenia

0. input: a set of training objects S and clearness tolerance t
1. build N evolutionary decision trees upon objects from S
2. copy all objects from S, classified in leaves in accordance with clearness tolerance t in at least $\lfloor N/2+1 \rfloor$ trees, to a set S^* and clear from a set S
3. from all N decision trees create M initial classification rules (all leaves in accordance with clearness tolerance t are used)
4. with the *proGenesys* system create another M classification rules randomly; initial population now contains $2 \times M$ classification rules
5. using the *proGenesys* system evolve the final set of rules for classifying objects from S^* at the current level
6. find the optimal final set of rules using simple optimization
7. if S is not empty (there are still some unclassified objects for the next level): 1) add $|S|$ randomly chosen objects from S^* to S; 2) increase clearness tolerance $t=2 \times t$; 3) repeat the whole procedure from step 1
8. finish if S is empty

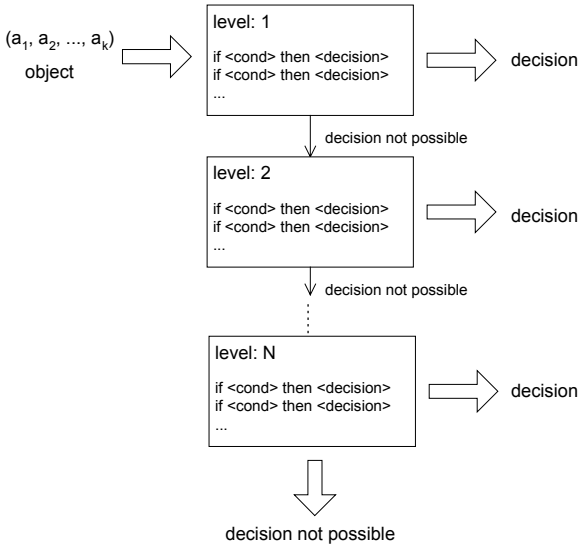

Figure 1. Multi-level hierarchical classification model

2.1 Genetic algorithm for the construction of decision trees

First step of the genetic algorithm is the creation of the initial population. A random decision tree is constructed based on the following algorithm:

0. input: number of attribute nodes N that will be in the tree
1. select an attribute X_i from the set of all possible attributes and set it as a rood node t
2. in accordance with the selected attribute's X_i type (discrete, continuous) define a test for this node t: 1) for continuous attributes in a form of $f_t(X_i) < \phi_i$, where $f_t(X_i)$ is the attribute value for a data object and ϕ_i is a split constant; 2) for discrete attributes two disjunctive sets of all possible attribute values are randomly defined

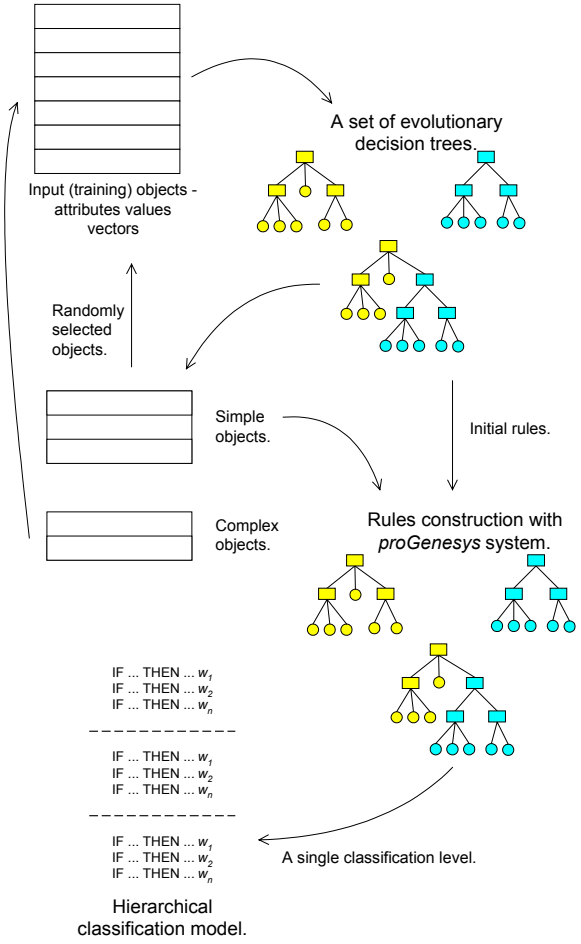

Figure 2. The diagram of AREX algorithm

3. connect empty leaves to both new branches from node t
4. randomly select an empty leaf node t (the probability of selecting an empty leaf is decreased with the depth of the leaf in a growing tree)
5. randomly select an attribute X_i from the set of all possible attributes (the probability of choosing an attribute depends on a number of previous uses of that attribute in a tree – in this manner unused attributes have better chances to be selected)
6. replace the selected leaf node t with the attribute X_i and go to step 2
7. finish when N attribute nodes has been created

For each empty leaf the following algorithm determines the appropriate decision class: let S be the training set of all training objects N with K possible decision classes d_1, .., d_K and N_i is the number of objects within S of a class d_i. Let S^t be the sample set at node t (an empty leaf for which we are trying to select a decision class) with N^t objects; N_i^t is the number of objects within S^t of a decision class d_i. Now we can define a function that measures a potential percentage of correctly classified objects of a class d_i:

$$F(t,i) = \frac{N_i^t}{N_i} \quad (1)$$

Decision d_i^t for the leaf node t is then set as a decision d_i, for which $F(t,i)$ is maximal.

The ranking of an individual DT within a population is based on the local FF:

$$LFF = \sum_{i=1}^{K} w_i \cdot (1 - acc_i) + \sum_{i=1}^{N} c(t_i) + w_u \cdot nu \quad (2)$$

where K is the number of decision classes, N is the number of attribute nodes in a tree, acc_i is the accuracy of classification of objects of a specific decision class d_i, w_i is the importance weight for classifying the objects of a decision class d_i, $c(t_i)$ is the cost of using the attribute in a node t_i, nu is number of unused decision (leaf) nodes, i.e. where no object from the training set fall into, and w_u is the weight of the presence of unused decision nodes in a tree.

2.2 System *proGenesys* for automatic evolution of programs

For constructing classification rules we developed a system for the evolution of programs in an arbitrary programming language, described with BNF productions – *proGenesys* (program generation based on genetic systems). In our approach an individual is represented with a syntax tree (a derivation tree), as it is usual for grammar-based GP [4,12]. To get the final solution this tree (genotype) is transformed into a program (phenotype) [11].

First step of automatic programming is the initialization of the initial population.. For the successful continuation of the program evolution process the initial programs should be evenly distributed [3]. Known program initialization procedures are theoretically correct but are working well only for small problems [6]. The problem is that limitations regarding the tree size are not considered during the induction. For this reason a lot of built trees have to be rejected, which is time consuming. An initialization procedure based on dynamic grammar pruning as proposed in [10] could solve the problem. In the *proGenesys* system a procedure is used that allows the induction of a program tree of exactly specified size (figure 3).

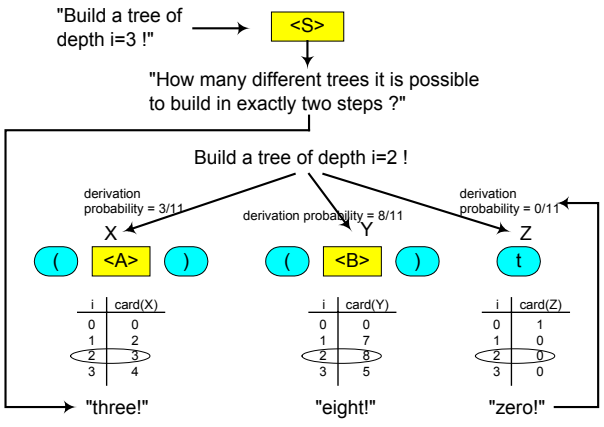

Figure 3. Controlled construction of derivation trees of a specific depth

A good classification rule should simultaneously be clear (most of the objects covered by the rule should fall into the same decision class) and general (it covers many objects – otherwise it tends to be too specific). Those two criteria can be measured with the following formulas:

$$generality = \frac{num.\ of\ classified\ objects - 1}{num.\ of\ objects\ of\ this\ decision\ class} \quad (3)$$

$$clearness = 1 - \frac{\frac{\omega_2}{\Omega_2}}{\frac{\omega_1}{\Omega_1}} \quad (4)$$

where ω_1 is number of objects covered by the rule that belong to the most frequent decision class, ω_2 is number of objects covered by the rule that belong to the second most frequent decision class, Ω_1 is number of all objects in the training set that belong to the most frequent decision class of the rule, and Ω_2 is number of all objects in the training set that belong to the second most frequent class of the rule. Now a fitness function can be defined as

$$FF = clearness \times generality + \sum_{i=1}^{N} c(t_i) \quad (5)$$

where the last part represents a cost of the use of specific attributes, the same as in the local fitness function *LFF* in building decision trees.

2.3 Finding the optimal set of rules

System *proGenesys* is used to evolve single rules, whereas for the classification of all objects on a specific level a set of rules is required. For this purpose between all the evolved rules a set of rules should be found that together classify all the objects – with high classification accuracy and a small number of rules. A problem is solved with a simple genetic algorithm that optimizes the following fitness function:

$$FF = \sum_{i=1}^{K} w_i \cdot (1 - acc_i) + \sum_{i=1}^{N} c(t_i) + w_m \cdot nm + w_u \cdot nu \quad (6)$$

where the first two parts are the same as in the LFF for building decision trees, and nm is the number of multiple classified objects, nu is the number of non-classified objects, and w_m and w_u are the corresponding weights. The appropriate coverage of the training set is thus achieved by reducing the number of not classified and multiple classified objects. A more advanced way to achieve the uniform coverage is discussed in [1].

3 ASSESSMENT OF RESULTS AND DISCUSSION

To make an objective assessment of our method a comparison of the obtained results has been made with the reference method for building decision trees, namely C5/See5. The following quantitative measures of efficiency have been used:

$$accuracy = \frac{num.\ of\ correctly\ classified\ objects}{num.\ of\ all\ objects} \quad (7)$$

$$sensitivity = \frac{tp}{tp + fn} \quad (8)$$

$$specificity = \frac{tn}{tn + fp} \quad (9)$$

and also measures of efficiency regarding the understandability of discovered knowledge: number of rules and number of used attributes. Experiments have been made with five real-world databases, four from the field of medicine and one from the field of software engineering.

3.1 MVP database

Prolapse is defined as the displacement of a bodily part from its normal position. The term mitral valve prolapse (MVP) [2], therefore, implies that the mitral leaflets are displaced relative to some structure, generally taken to be the mitral annulus. The silent prolapse is the prolapse which can not be heard with the auscultation diagnosis and is especially hard to diagnose. The implications of the MVP are the following: disturbed normal laminar blood flow, turbulence of the blood flow, injury of the chordae tendinae, the possibility of thrombus' composition, bacterial endocarditis and finally hemodynamic changes defined as mitral insufficiency and mitral regurgitation. Mitral valve prolapse is one of the most prevalent cardiac conditions, which may affect up to five to ten percent of normal population and one of the most controversial one.

Using the Monte Carlo sampling method 900 children and adolescents were selected representing the whole population under eighteen years of life. Routinely they were called for an echocardiography no matter of prior findings. 631 of them passed an examination of their health state in a form of a carefully prepared protocol specially made for the syndrome of MVP. The protocol consisted of 103 parameters that can possibly indicate the presence of MVP. Distribution of the three decision classes were: 5% prolapse, 6% silent prolapse, and 89% ok. A summary of the MVP database is presented in table 1.

3.2 MRACID database

Through the help of classical medical research it has been established that surgeries under the general anesthesia cause in organism a tendency to dropping the blood's pH value, also known as predisposition to acidemia. The results of blood's gases analysis, serum electrolytes analysis, blood count and values of length of the operation, blood pressure, pulse, temperature, age, weight, height, sex, duration of the surgery and transfusion volume have been used to define altogether 25 attributes. Altogether 82 children patients were thoroughly examined in the study. Distribution between the two decision classes were: 88% with the tendency towards acidosis and 12% without the tendency towards acidosis. A summary of the MRACID database is presented in table 1.

3.3 AAP database

The early and accurate diagnosis of acute appendicitis is a difficult and challenging problem in everyday clinical routine. Of major concern are the perforation rate (up to 20%) and negative appendectomy rate (up to 30%). An important factor in the error rate is poor discrimination between acute appendicitis and other diseases that cause acute abdominal pain (AAP) [7]. This error rate is still high, despite considerable improvements in history-taking and clinical examination, computer-aided decision-support and special investigation, such as ultrasound.

The database was built-up during an Concerted Action funded by the European Commission during the COPERNICUS program. Data was collected in 16 centers from Central and Eastern Europe. For data collection the computer program developed in the MEDWIS program was used. Medical terminology was translated into 10 different languages, so that the participating centers could be provided with national versions of the software. The final diagnosis was based on the diagnosis at discharge. A summary of the AAP database is presented in table 1.

3.4 AV database

Continuous time signals, like EEG, are often used in medicine by the medical staff to provide some information about the state of the patients. The objective of this study was to recognize patients with Alzheimer disease based on the recorder EEG signals. EEG or electroencephalogram is a record of electrical activity in the human brain (Figure 4). The EEG is one of the tools a neurologist uses in the diagnosis of brain conditions. There are several methods to analyze continuous time signals in medicine. Data mining techniques and data-based classification methods are not appropriate to deal with these kind of data. However, applying some domain knowledge enable one to extract parameters from EEG signals, that provide enough distinguishable information about a specific signal (Figure 5). In this way a set of continuous attributes are obtained, based on the domain knowledge, which can be used to construct a decision tree (avg, min, max, median in the frequency domain, avg, median, max, balanced median in the time domain, etc.). Distribution between the two decision classes was very unbalanced: 88% of patients with Alzheimer disease and 12% with some other vascular problems. A summary of the AV database is presented in table 1.

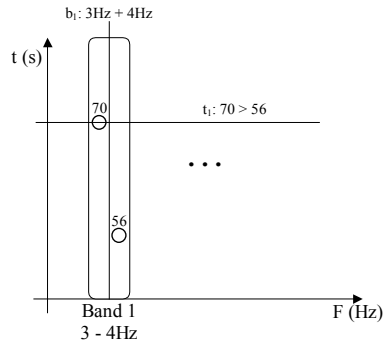

Figure 4. Selection of attributes from an EEG signal (band 1), showing the characteristic points for the frequency (b1) and time (t1) domain

3.5 SCM database

Software development is a complex and complicated process in which software faults are inserted into the code by mistakes during the development process or maintenance. It has been shown that the pattern of the faults insertion phenomena is related to the

measurable attributes of software objects, especially with the software metrics [9].

In the study a medical software system has been used, developed at the University of Udine, Italy. It consists of 904 software modules containing over 2 million lines of code in C++ language. For all modules a set of metrics' results measured after the development of the system and a seven years of maintenance reports about found errors exist. Based on the number of discovered errors modules have been tagged either as OK or dangerous. The aim of the study is to classify the modules in those two groups based on the metrics' results. A summary of the SCM database is presented in table 1.

Table 1. A summary of tested databases

dataset	Δ	∇	◊		
mvp	361	102	5,23%	5,71%	89,06%
mracid	82	24	87,80%	12,20%	
aap	3439	18	47,34%	52,66%	
av	207	40	88,00%	12,00%	
scm	904	168	23,56%	76,44%	

Δ num. of objects ∇ num. of attributes ◊ distribution of decision classes

3.6 The results

When using evolutionary methods it is important to guarantee the convergence of evolutionary runs. Figure 5 shows average convergence rates of 10 independent evolutionary runs for each database for the first 100 generations.

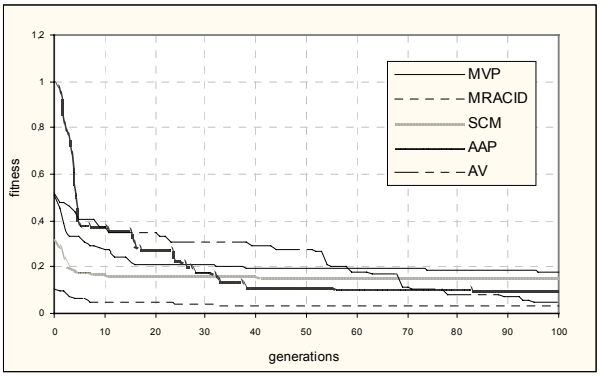

Figure 5. Fitness scores of the best solutions in each population

For the objective comparison of efficiency for a knowledge discovery method upon each database the universal criteria have been defined:

$$efficiency_{classification} = sensitivity \times specificity \quad (10)$$

$$simplicity = 1 - \frac{average\ number\ of\ rules\ for\ a\ method}{max.\ number\ of\ rules\ for\ all\ methods} \quad (11)$$

$$efficiency = sensitivity \times specificity \times simplicity \quad (12)$$

All three criteria are normalized values between 0 and 1, therefore the result (i.e. efficiency of a method) is expressed as a value in percents between 0 and 100%, where the most efficient method has the highest score. With those criteria efficiency of our method can be objectively evaluated upon all five databases (table 2); it has been compared with the C5/See5 tool [13].

Table 2. A comparison of efficiency of methods C5/See5 and AREX

	C5/See5			AREX		
dataset	Δ	∇	◊	Δ	⊗	◊
mvp	92,28	10,59	5,53	91,47	41,18	31,17
mracid	91,94	25,71	12,86	95,98	31,43	27,34
aap	84,84	23,69	17,09	83,25	95,04	66,00
av	88,66	13,75	7,14	93,07	31,25	23,07
scm	79,16	20,32	8,77	83,27	90,97	58,94

Δ accuracy ∇ simplicity ◊ efficiency

To be able to confirm a higher efficiency of our method AREX when compared to C5/See5, the statistical significant difference between the results has been calculated using the Leven-T test for the comparison of equivalence of variances. The results show statistical significant difference with high probability for sensitivity (level 0.002), efficiency (0.001), and simplicity (0.000). Regarding all the results achieved with our method AREX we can say, that automatic programming is an appropriate method to achieve more efficient knowledge discovery on some real-world medical databases.

REFERENCES

[1] C. Anglano, A. Giordana, G. Lo Bello, L. Saitta, *An experimental evaluation of coevolutive concept learning*, 19-27, International Conference on Machine Learning ICML98, 1998.

[2] R. Devereoux, 'Diagnosis and Prognosis of Mitral Valve Prolaps', *The New England Journal of Medicine*, **320**, 1077-1079, (1989).

[3] A. Geyer-Schulz and W. Böhm, *Exact uniform initialization for genetic programming*, Foundations of Genetic Algorithms 4, 1996.

[4] F. Gruau, *On using syntactic constraints with genetic programming*, 377-394, Advances in Genetic Programming II, MIT Press, 1996.

[5] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, Inc., 2000.

[6] H. Hörner. A C++ class library for genetic programming. M.Sc. thesis, Vienna University of Economics, 1996.

[7] Ch. Ohmann, V. Moustakis, Q. Yang, K. Lang, 'Evaluation of automatic knowledge acquisition techniques in the diagnosis of acute abdominal pain', *Artificial Intelligence in Medicine*, **8**, 23-36, (1996).

[8] V. Podgorelec and P. Kokol, *Evolutionary decision forests – decision making with multiple evolutionary constructed decision trees*, 97-103, Problems in Applied Mathematics and Computational Intelligence, WSES Press, 2001.

[9] D.J. Paulish and A.D. Carleton, 'Case Studies of Software Process Improvement Measurement', *IEEE Computer*, **9**, 50-57, (1994).

[10] A. Rattle, M. Sebag, *Genetic programming and domain knowledge: beyond the limitations od grammar-guided machine discovery*, 211-220, Parallel Problem Solving from Nature VI, Springer Verlag, 2000.

[11] C. Ryan, *Shades – a polygenic inheritance scheme*, 140-147, Proceedings of Mendel'97 Conference, 1997.

[12] P.A. Whigham, 'Inductive Bias and Genetic Programming', *IEE Conference Proceedings*, **414**, 461-466, (1995).

[13] –, RuleQuest Research Data Mining Tools, http://www.rulequest.com