rule extraction, evolutionary computation

Halina KWAŚNICKA*,
Urszula MARKOWSKA–KACZMAR*,
Tomasz OSOJCA*

EVOLUTIONARY APPROACH TO RULE EXTRACTION FROM MEDICAL DATA

In the paper the method called CGA based on a cooperating genetic algorithm is presented. The CGA is developed for searching a set of rules describing classes in classification problems on the basis of training examples. The details of the method, such as a schema of coding (a chromosome), and a fitness function are shortly described. The method is independent of the type of attributes and it allows choosing different evaluation functions. Developed method was tested using different benchmark data sets. Next, in order to evaluate the efficiency of CGA, it was tested using the Breast Cancer data set with 10 fold cross validation technique.

1. INTRODUCTION

Nowadays people dispose terabytes of collected data. Existing databases are potentially large sources of useful knowledge, but - to be useful - this concealed knowledge must first be drawn out from databases in the form comprehensible for people. This process is called data mining. One of its tasks is classification. Let us assume that we have a set of m objects represented by a vector $x_i = [x_{i,1}, x_{i,2}, .., x_{i,k}]$, where: $i=1, .., m$. Classification can be defined as a splitting the objects into mutually disjoined v categories. In some domains it is required to acquire rules specifying the principles of classification. Such rules can be very useful in medicine.

Many medical problems can be represented as tasks of searching large spaces: a pathologist must search the space of all possible cell features to make a diagnosis, a radiologist must search a whole space of possible therapies to plan a sequence of radiotherapy, etc. Usually, search spaces are very large and the tasks are difficult. The useful techniques are, among others, evolutionary algorithms (classification problems, rule discovery, planning, etc, e.g. [5-6, 9-10]). Extended bibliography concerning EA in medicine one can find in [7-8].

The formal form of a classification rule is presented by:

$$IF\ prem_1\ and\ prem_2\ and\ ...\ and\ prem_k\ THEN\ class_b$$

* Department of Computer Science, Wroclaw University of Technology, [halina.kwasnicka, kaczmar]@pwr.wroc.pl

Each premise $prem_i$ imposes a constraint on the single input variable: for variables with real values premise defines a range $[a_i ; b_i]$, for enumerate – it assigns a specific values. A rule is activated when considered example satisfies requirements coded in the premises. The paper presents developed method of rule discovery from data, called CGA (Cooperating Genetic Algorithm). In the following sections the details of this method are described. The results of experiments studying its efficiency are also presented and discussed.

2. THE CGA METHOD

2.1. PROBLEM DEFINITION

The rule extraction problem can be treated as a task of searching for the set of classification rules on the basis of training examples in order to find rules that in the best way represent the dependency hidden in the explored data. This problem can be seen as an optimization problem. If we take into account the existence of different type of attributes (enumerative, binary, continuous and discrete) in one training set, the need to find rules describing different classes, and the huge dependencies between attributes, we obtain the task, which is similar to *NP* problem. The above observations lead to the result that evolutionary method can be useful in this task. The idea is not new [3]. Unfortunately, the level of complexity of this problem prevents the application of a simple genetic algorithm (GA), so existing methods applying a genetic algorithm differ in the way of coding and obtaining the final set of rules [3-4].

2.2. THE IDEA

To find the set of rules describing classification is a multimodal problem. A genetic algorithm usually tends to the global optimum. It means that a GA, working with a single rule encoded in one chromosome, assures one the best rule, which characterizes one class. To obtain rules for different classes we implemented as many subpopulations as many classes exist in the given problem (Fig. 1). It means that the label of subpopulation denotes the class. In this case, there is no need to encode the conclusion representing class in the chromosome. The problem of searching the set of rules in one subpopulation still exists, because it is very rarely possible that by one rule we could describe all examples in one class. This problem is solved by implementation the niching method.

In our method, the description of each subpopulation is based on hierarchical GA, the first one works in $Level_1$ – it is responsible for selection of active attributes for the rule construction. The second one, working in $Level_2$ searches for the boundaries of ranges for real valued attributes and values for other attributes (with binary or enumerative value). The algorithm (for each subpopulation) goes as follow (Fig.2):

1. Random generation of initial population in the $Level_1$
   {binary chromosomes contain *n* flags for attributes, *n* – the number of attributes, *one* in an individual indicates an active premise (active attribute) in the evolved rule}
2. For each individual in the population perform evolution (*v* generations) in the $Level_2$
   {chromosomes in $Level_2$ contain a boundaries for real valued attributes and values for attributes with binary or enumerate values}
3. Evaluate individuals in the $Level_1$ taking into account an effect of evolution in $Level_2$
4. Check the stop conditions, if not:
5. Select individuals to the reproduction
6. Perform mutation and crossover and go to the 2.

For each individual from the population in $Level_1$ evolution in the $Level_2$ searches for the best values of active attributes in the premises of the evolved rule. Evolution in this level operates as follow:

1. Create initial chromosomes (individuals) {take into account allowed values for particular attributes, Fig. 3}
2. Evaluate each individual
   - Decode a chromosome into a rule according to flags in the relevant individual in $Level_1$
   - Evaluate obtained rule taking into account examples belonging to the ascribed class and defined fitness function (see next sections)
3. Check for stop conditions, if not:
4. Select individuals for reproduction
5. Perform mutation and crossover, the operators depend on the type of modified attribute and go to the 2.

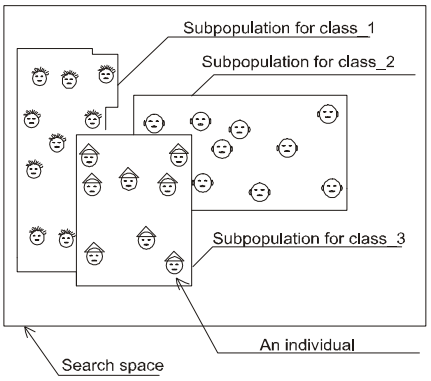

Fig. 1. An example population with 3 subpopulations for 3 classes

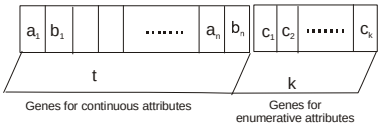

Fig. 2. The scheme of a chromosome in the $Level_2$ of CGA method

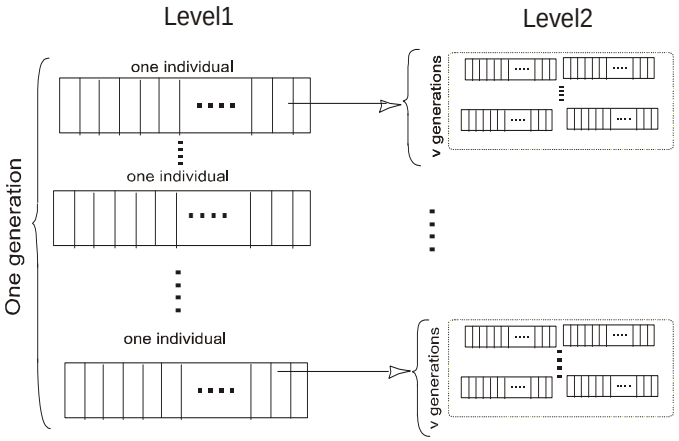

Fig. 3. The scheme of cooperating genetic algorithms in CGA method searching for rules for one class

2.3. THE FORM OF CHROMOSOMES AND GENETIC OPERATORS

In developed CGA we use different chromosomes and genetic operators in both levels. Chromosomes in $Level_1$ consist of *n* binary values (*n* is the number of premises in the rule, simultaneously it is a number of attributes in examples). The standard genetic operators are

used in $Level_1$. The roulette wheel selection is realized taking into account evaluation of individuals.

In $Level_2$ an individual consists of the two parts. The first one concerns real valued attributes. For each attribute in this case, a chromosome contains a pair of real values – the boundaries of its range. The second part of chromosomes contains values for all other attributes (i.e., enumerative ones). So we can say, that we consider two types of genes: continuous and enumerate (Fig. 2). Mutation process differs for the first and the second parts of chromosomes. Mutation of boundaries of ranges works as in evolutionary strategies. Values of x after mutation are calculated according to the equation (1):

$$x(t+1) = x(t) + N(0,\delta) \tag{1}$$

where $N(0,\delta)$ is a Gaussian number, deviation δ usually evolves as a part of chromosomes. Such mutation is very natural because the small changes occur more frequently then the big ones. The crossover operator is realized as an average value of the parent vectors.

2.4. FITNESS FUNCTIONS

Designing a fitness function is one of the essential tasks in the application of a genetic approach. In the paper different fitness functions are tested. We define one basis, which concerns only one objective, namely *accuracy*. This form can be enhanced by incorporating other objectives, which have to be satisfied during the rule extraction process. In this way we developed two other forms of fitness function. {This sentence is not clear enough and ought to be corrected} In this case the fitness function it is expressed by the number of correctly classified patterns (*TP* – True Positive), to the number of wrong classified examples (*FP* – False Positive). Formally, it is shown by (2).

$$fit = \begin{cases} \dfrac{TP}{FP} \; when \; FP > 0 \\ 2 * TP \; when \; FP = 0 \end{cases} \tag{2}$$

where *TP* is the number of examples from training set, which attributes meet the conditions in the premises and their class is suitable to the one specified by subpopulation; *FP* is the number of examples, which satisfy the premises, but they belong to other class.

This fitness function promotes individuals, which cover examples from the correct class by multiplying the value *TP* by 2 in the case when *FP* is equal to 0. The second criterion in the extracting rules is *comprehensibility* for human. It can be measured by the number of premises in the rule. This criterion also ensures the extraction of rules, which are more general. The modification fit_C of fitness function satisfying this criterion is expressed by (3).

$$fit_C = fit - \left(\beta * \frac{NoP}{NoA} \right) \tag{3}$$

where *fit* is expressed by (4); β is the parameter set by the user; *NoP* is the number of premises in the rule; *NoA* is the number of attributes in the example.

The modification (3) relies on the making allowance of a penalty, proportionally to the number of active premises. The role of β is to tune the value of a penalty in this way that it introduces proper change of basic fitness function *fit*. The next change in fitness function resulting from the experience acquired on the base of primary experiments relies on the reward of the rule for the rarity. Especially at the end of the search, the rule which covers one or more new examples (uncovered yet) is very valuable for the final effect. The part of fitness function rewarding an individual for the rarity is described by (4).

$$R = \begin{cases} \sum_{i=1}^{T} \frac{1}{NC_i} \; for NC_i > 0 \wedge example\ x_i\ satisfies\ evaluated\ rule \\ 0 \quad for\ NC_i = 0 \end{cases} \qquad (4)$$

where T is the total number of examples in the given class covered by the evaluated individual; NC_i is the number of other individuals in the population, which cover i-th example.

The meaning of this modification is such that the rule gets a reward for each covered example. It will be inversely proportional to the number of rules, which cover the example. The fitness function fit_R respecting the element promoting for rarity is shown in (5).

$$fit_R = fit + \alpha * R \qquad (5)$$

where *fit* is the basic fitness function assigned? by 4?; R is the part expressing reward for the rarity defined by 6?; α is the coefficient set by the user.{It worth to mention what the numbers 4 and 6 mean}

The offspring-rule covering an example, which was not covered by other individuals in the population, is additionally awarded during the comparison with its parents. It obtains an award equal to a half of maximum value of fitness function for a given class, what guarantees a presence of this individual in the population according to principle of preselection. It substitutes the worse parent. To avoid a domination of very good individuals a scaling of the fitness function was introduced, which is realized with the application a linear function.

Presented above fitness functions are used for individuals' evaluation in $Level_2$. Because an evolution in $Level_2$ is made for each individual in $Level_1$, as a fitness value of the individual in $Level_1$ we take a fitness of the best evolved individual in $Level_2$. The roulette wheel selection is used in $Level_1$.

Our main objective is to find the set of rules for each class, covering as many as possible examples. Our approach to this problem is discussed in the next section.

2.5. THE FINAL SET OF RULES

In order to obtain as small as possible the final set of rules, in the whole population the rule that covers the most of the new examples is chosen {It is necessary to correct this sentence} . "New examples" means those examples, which are not covered by rules, which are the member of the final set of rules, already. All rules that belongs to the final set of rules have to satisfy the condition that they do not cover false examples (FP =0). As the first one, the rule that covers the biggest number of examples is incorporated to the final set of rules. Next, the rules that cover the biggest number of examples, uncovered yet, are inserted there. When there are two rules with the same number of the covered examples (uncovered yet), the rule with the less number of premises is chosen. The above described algorithm is presented in a formal form as follows:

```
While there are uncovered examples in the training set and there is a rule covering almost one of these examples
        choose the rule with the biggest number of examples
        add this rule to the final set of rules
        mark the new covered examples as covered
End {while}
```

3. EXPERIMENTAL STUDIES

The developed method was tested in two phases. First, we evaluated the influence of fitness function on a course and results of evolution. In all presented experiments benchmark data sets from [1] were applied. The results of the tests made with *Iris*, *Breast Cancer* and *Heart* data sets are presented in Tables 1., 2. and 3.

Let us look at the Table 1. In the third column, sign '–' represents the option when the evolution was stopped after assumed number of generations, while X means that tests were stopped when the accuracy achieved 100%. This high accuracy relates to the bigger number of premises and the bigger number of rules. The tests confirm the appropriate influence of modified fitness function for an evolution process. It can be noticed that fitness function modified by complexity fit_C causes decreasing of the number of premises but the accuracy is also smaller. Fitness function with rarity fit_R gives better accuracy comparing to others. We combine both modifications in the fitness function fit_{CR} and the improvements in both objectives can be observed. The same tendency can be observed on the base of experiments on *Heart* and *Breast Cancer* data sets (see Table 2. and 3.)

Table 1. The results of experiments for *Iris* data set

Fitness	Number of induviduals	Stop	Average accuracy (%)				*Average number of rules*				Average number of premises			
			Setosa	*Versicolour*	*Virginica*	Total	*Setosa*	*Versicolour*	*Virginica*	Total	*Setosa*	*Versicolour*	*Virginica*	Total
fit	50	X	100	99	100	98,6±0,6	1,4	2,2	2,9	2,1±0,8	1,2	2,4	1,6	1,7±0,3
	50	-	100	98	98	97,6±1,1	1	1,3	2,1	1,4±0,5	1,1	2,4	1,2	1,5±0,2
fit_C	20	X	100	100	99	98,4±0,4	1,3	2,3	2,9	2,1±0,4	1,3	2,4	1,5	1,7±0,3
	50	-	100	98	98	97,6±1,1	1	1,2	2,1	1,4±0,3	1	2,3	1,1	1,5±0,3
fit_R	10	X	*100*	*100*	*100*	*100*	1,8	2,4	3,3	2,5±0,5	1,4	2,2	1,7	1,8±0,4
	10	-	*100*	*99*	*99*	*98,2±1,2*	1	1,9	2,7	1,8±0,5	1,1	2,4	1,5	1,6±0,4
fit_{CR}	10	X	100	100	100	100	1,2	2,5	3,2	2,3±0,4	1,3	2,2	1,7	1,7±0,4
	50	-	**100**	**100**	**100**	**100**	1	1,5	2,8	1,6±0,3	**1**	**2,2**	**1,5**	**1,5±0,3**

Table 2. The results for the *Breast Cancer data set*

Fitness	Number of individuals	Average accuracy (%)			*Average number of rules*			Average number of premises		
		Benign	*Malignant*	Total	*Benign*	*Malignant*	Total	*Benign*	*Malignant*	Total
fit	200	99	95	97,6±0,4	14	14	14±1,7	3,4	2,2	2,8±0,1
fit_C		99	94	97,2±1	15	13	14±1,9	2,9	2	2,45±0,1
fit_R		*99*	*99*	99±0,6	15	21	18±2	3,4	2,4	2,9±0,1
fit_{CR}		**98**	**99**	98,3±0,5	15	21	18±1,5	**2,8**	**2,4**	2,5±0,1

Table 3. The results for the *Heart* data set

Fitness	Number of individuals	Average accuracy (%)			*Average number of rules*			Average number of premises		
		Absence	*Presence*	Toatal	*Absence*	*Presence*	Total	*Absence*	*Presence*	Total
fit_I	200	90	90	90±1	26	21	23,5±1,3	4,5	3,9	4,2±0,1
	400	94	94	94±1,2	24	21	22,5±1	4,7	4,2	4,4±0,2
fit_{IZ}	400	92	92	92±0,7	23	20	21,5±1,4	3,5	3	3,2±0,2
		94	94	94±1	24	21	22,5±1,1	4,3	3,9	4,1±0,1
fit_{IU}		*98*	*97*	*97,5±1,5*	27	25	26±1,5	4,7	4,3	4,5±0,2
fit_{IZU}		**98**	**97**	**97,5±2**	27	23	25±1,6	**4,4**	**4,1**	**4,2±0,1**

Table 4. The results of experiments for *Animals* data set with initial population created on the basis of examples

Fitness	Number of individuals	Stop	Average accuracy (%)	*Average number of rules*					Average number of premises				
				Giraffe	*Dog*	*Cat*	*Horse*	Total	*Giraffe*	*Dog*	*Cat*	*Horse*	Total
fit_1	20	X	100	5	7	4	5,3	5,3±0,4	31	32	31	31	31,2±1
		-	100	1,3	1,3	1,3	1	1,2±0,1	24	30	32	34	29,5±1,5

The *Animals* data set contains 72 attributes. Such a big number of attributes causes that in the initial population there were only individuals with fitness function equal to 0. Even for a very large population (thousands individuals) this phenomena was observed. To resolve this problem the generation of initial population on the basis of training data set was introduced.

Table 5. The results of experiments for the *Animals* data set with an initial population generated on the basis of examples and inactive premises

Fitness	Number of individuals	Stop	Average acccuracy (%)	*Average number of rules*					Average number of premises				
				Giraffe	*Dog*	*Cat*	*Horse*	Toatal	*Giraffe*	*Dog*	*Cat*	*Horse*	Toatal
fit_1	20	X	100	1	1	1	1	1	2,7	1,7	1,7	6	3,1±0,2
		-	100	1	1	1	1	1	1	1	1	1	1

The results are shown in the Table 4. The number of premises is relatively large. It is the result of the creation of the initial set of population, where each example becomes particular rule. To avoid this problem additionally we made an experiment setting all premises as inactive. This option joined with a relatively small number of mutated genes gives gradually improvement; the results are shown in Table 5. For each class only one rule was found, it indicates easy separation of classes. Similar experiments were performed with *Iri*s, *Breast Cancer* and *Heart* data sets. In this case the results were not interesting, because the accuracy was equal to 0. In order to compare the obtained results with other methods described in [2], we used the 10 fold cross validation technique.

That comparison is shown in Table 7. It presents the accuracy in % made on the basis of *Breast Cance*r. The accuracy for CGA is equal to 93 %, which is less than MLP+BP or NB or LDA because they have more than 96% [2]. This phenomenon in CGA can be caused by the applied mutation in $Level_2$.

Table 6. The results of experiments for *Iris* and *Breast Cancer* data set with 10 fold cross validation

Fitness	Accuracy - training (%)	Accuracy – testing set (%)
Iris		
fit	97	94.2
fit_Z	94	91
fit_U	99	92
Breast Cancer		
fit	97	93
fit_Z	94	91
fit_U	99	92

Table 7. The comparison of CGA with other rule extraction methods, using 10 fold cross validation (*Iris* data set)

Method	10-fold-cross-validation (%)
MLP+BP	96.7
NB - naive Bayes	96.4
LDA - linear discriminant analysis	96.0
C 4.5 tree	94.7±2.0
CGA	**93**
C 4.5 rules	86.7±5.9

The mutation in the second level (causing the small changes) finds the boundaries separating classes which overfit the training examples. One of the solutions could be a modification of fitness function, which would reward individuals containing genes with wider ranges. Results presented in Table 6 show that fitness function with rarity increases accuracy with training data sets but decreases it when testing data set is used. It can be perceived as the overtraining effect, which gives better accuracy with the training set but simultaneously it gives the lost of generality of the final set of rules. The same tendency can be notice with other fitness function.. These observations lead to the rejection of fitness function modified by rarity in real applications. In the developed method with the fitness function *fit* mutation derived from evolutionary strategies assures high accuracy for training examples but it does not produce more general rules. Probably, the results would be better with modified fitness functions. It is worth to notice that the comparison was possible for one data set, so the conclusion can not be general.

4. CONCLUSIONS

The novelty of the rule extraction method CGA is based on the combination of the genetic algorithm and a kind of evolutionary strategy. The method is able to extract rules describing the classified examples. Different fitness functions were tested. In the experiments the best results were achieved with modified fitness function, which rewards individuals for rarity and complexity. Comparing results given by CGA with other methods we used in the experiments only simple fitness function. To objectively evaluate the proposed method the further experiments for other data sets are necessary. However, even with this simple fitness function, the accuracy equal to 93 % is the satisfying solution. CGA was used as a method of rule extraction on the basis of a set of examples with known class-membership.{Last two sentences should be cancelled or corrected}.

BIBLIOGRAPHY

[1] BLAKE C.C., MERZ C.: *UCI Repository of Machine Learning Database*s, University of California, Irvine, Department of Information and Computer Sciences, 1998. [1]

[2] DUCH W., ADAMCZAK R., GRABCZEWSKI K.: *A new methodology of extraction, optimization and application of crisp and fuzzy logical rule*s, IEEE Transactions on Neural Networks 12, p. 277-306, 2001. [2]

[3] FIDELIS M.V., LOPES H. S., FREITAS A.A.: *Discovering comprehensible classification rules with genetic algorith*m, Proc. Congress on Evolutionary Computation (CEC-2000), p. 805-810, La Jolla, July 2001. [4]

[4] FRANCISCI D., BRISSON L., COLLARD M.: *A scalar evolutionary approach to rule extractio*n, Rapport de recherche, ISRN I3S/RR-200312-FR, 2003. [3]

[5] HOLMES JH, DURBIN DR, WINSTON FK. The learning classifier system: an evolutionary computation approach to knowledge discovery in epidemiologic surveillance. *Artificial Intelligence in Medicine* 2000; 19(1):53-74.

[6] KOMOSIŃSKI M, KRAWIEC K. Evolutionary weighting of image features for diagnosing of CNS tumors. *Artificial Intelligence in Medicine* 2000; 19(1):25-38.

[7] KWAŚNICKA H., Obliczenia ewolucyjne w medycynie. W: „Kompendium informatyki medycznej". Red. Radosław Zajdel [i in.]. Bielsko-Biała, Alfa Medica Press, cop. s. 365-402, 2003.

[8] PEÑA-REYES CA, SIPPER M. Evolutionary computation in medicine: an overview. *Artificial Intelligence in Medicine* 2000; 19(1):1-23.

[9] VINTERBO S, OHNO-MACHADO L. A genetic algorithm approach to multi-disorder diagnosis. *Artificial Intelligence in Medicine* 2000; 18(2):117-132.

[10] YU Y, ZHANG JB, CHENG G, SCHELL MC, OKUNIEFF P. Multi-objective optimization in radiotherapy: applications to stereotactic radiosurgery and prostate brachytherapy. *Artificial Intelligence in Medicine* 2000; 19(1):39-51.

Urszula Markowska–Kaczmar would like to thank Polish Committee for Scientific Research for partial financial support of research presented in the paper under grant number 4 T11 E 02323.