BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **7**, No 2

Sofia • 2007

An Analysis of Applicability of Genetic Algorithms for Selecting Attributes and Examples for the Nearest Neighbour Classifier

Gennady Agre[1], *Parvan Gioshev*[2]

[1] *Institute of Information Technologies,1113 Sofia*
E-mail: agre@iinf.bas.bg
[2] *Faculty of Mathematics and Informatics, Sofia University*
E-mail: parvan_83@abv.bg

***Abstract**: The paper discusses the problem of applicability of genetic algorithms for selecting examples and attributes for the Nearest Neighbour classifier. The emphasis is on the importance of the proper choice of the fitness function and the method for constructing new generations. Some improvements to the existing algorithms are proposed and their fruitfulness is experimentally proved on sixteen benchmark datasets. The empirical evaluation has shown that the increase in the classification accuracy of the modified algorithm is statistically significant.*

***Keywords:** Genetic algorithms, nearest neighbour classifier, feature selection.*

1. Introduction

The classical nearest neighbour (NN) algorithm is a very simple and effective classifier. Being initially developed for the purposes of pattern classification [7], it then has been used as a base for implementing several powerful case-based classification systems [12]. The algorithm avoids a lack of background knowledge by storing all training examples as an extensional description of the target categories (classes) and classifies an unseen example by the class of the nearest training example. The main problems with NN are its high classification cost and strong dependence of its accuracy from the number of irrelevant attributes. It has been shown, that the number of examples that have to be stored to achieve certain classification accuracy grows exponentially in the number of irrelevant attributes. Several experimental studies (e.g.[3]) are consistent with this result.

The first problem is often attempted to be solved by pre-processing the training examples aiming at generating a new example representation facilitating more efficient search. An example of such representation is *k- d* trees ([8]), that are able to reduce the classification cost to $O(\log_2 n)$, where n is the number of training examples. *k–d* trees have inspired the development of a set of new powerful point access and spatial access methods that are successfully used in advanced database systems [19].

The prototype-based learning (or prototype selection methods) is another approach for solving the problem with high classification cost of the NN method. The main idea is to reduce the size of stored examples by selecting only such of them (called “prototypes”) that represent the class in “the best” way. That means that the classification accuracy of such reduced or edited dataset should be (at least) comparable with the classification accuracy of the original database (i.e. containing all training examples). The nearest neighbours methods that use for classification only a part of training examples are often called edited or reduced nearest neighbour classifiers.

Prototype selection methods can be split on redundancy reduction, noise removal and hybrid algorithms combining previous two ones. The redundancy reduction algorithms aim at eliminating training examples that do not contribute to classification competence (e.g. examples from the interior of a densely packed class cluster). These algorithms can produce significant training set reduction but tend to preserve noise examples as apparent exceptions. In contrast, noise removal algorithms aim at eliminating noisy examples from the training set. These algorithms tend to a less dramatic size reduction (depending on the level of training set noise) and can also mistakenly remove important exceptional examples that are difficult to distinguish from the true noise. A comprehensive review of the prototype selection methods can be found in [1].

The task for selecting relevant attributes (or features) is to find such a subset of the original attributes that allows an inductive algorithm, running on data described by only these attribute subset, to generate a classifier with as highest as possible accuracy [9]. There are a number of different approaches to attribute subset selection that can be roughly spit on the filter approach and wrapper approach. The filter methods select features using a pre-processing step. They do not take into account the biases of the induction algorithms and select feature subsets independently of the inductive algorithms used. The best examples of such approaches are FOCUS [4], ReliefF [10] and feature filtering using decision trees [6]. In the wrapper approach, the feature subset selection is done using the induction algorithm as a black box. The wrapper conducts a search for a good feature subset using the induction algorithm itself as a part of the evaluation technique. The main problem of the wrapper approaches is that they are too slow since exploit very intensively the cross-validation procedure [9].

The task of prototypes and attribute selection may be also seen as a search problem and this point view is the basis for some techniques applying genetic algorithms (GA) to reduce the size of the NN classifier. In the approach of [14] the length of the chromosome is made equal to the size of the training set, and each allele is represented by Boolean number indicating whether the associated example is used or not. The authors in [11] have extended this approach – in their solution each bit of a binary chromosome represents one of the original training examples and attributes. It is obvious that both approaches are impractical in domains with many training examples and attributes.

A more scalable approach [16, 17] is to use a variable-length encoding. In this approach each allele contains an integer that points to a training example. It is worth to mention that this approach allows to reduce not only the size of the training set, but to remove irrelevant attributes as well. Since it is reported that the algorithm has achieved comparatively good results in both directions we have decided to analyze it in more details in order to understand the reasons of its success. The present paper proposes the detailed analysis of this approach and suggests some directions for its improvement.

The structure of the paper is as follows: the next section is devoted to a detailed description of the approach proposed by Rozsypal and Kubat. Section 3 describes our modifications to the mentioned above algorithm. Section 4 considers the experimental evaluation of the modified algorithm on fifteen benchmark data sets, and the last section is conclusions and our plan for future research.

2. Genetic algorithm of Rozsypal and Kubat

Before describing the details of the GA proposed by Rozsypal and Kubat, we will briefly present a general idea and structure of a genetic algorithm.

2.1. A general genetic algorithm

Genetic algorithms are intended for searching a space of possible solutions to identify the best one. The "best" solution is defined as the one optimizing a predefined numerical measure called the solution *fitness*. Although different implementations of the genetic algorithms vary in their implementation details, they usually share the following structure [13]:

- The algorithm works by iteratively updating a set of possible solutions, called population. On each iteration, all members of the population are evaluated according to the fitness function.
- A new population is generated by probabilistically selecting the most fit individuals from the current population.
- Some of the selected individuals are carried forward into the next population intact. Others are used as the basis for creating new offspring individuals by applying genetic operations such as crossover and mutation.

The described above genetic algorithm performs a randomized, parallel beam search for solutions that perform well according to the specified fitness function.

2.2. Description of the technique

A main element of a genetic algorithm is the population, which consists of so called *specimens* – alternative solutions of the problem to be solved. The authors in [17] represent each specimen by two *chromosomes* – the first one encodes training examples to be stored and used for classification and the second – example attributes to be used for calculating distances between a test example and stored training examples. In other words each *allele* (an element of the chromosome) contains an integer that points to a training example or to an attribute – in GA literature such encoding is usually referred to as value encoding.

The distance $d(\boldsymbol{x}, \boldsymbol{y})$ between a stored training example $<\boldsymbol{x}, c>$, $\boldsymbol{x} = \{x_{i1}, \ldots, x_{ip}\}$ and a training example $\boldsymbol{y} = \{y_1, \ldots, y_n\}$, $i_k \in \{1, \ldots, n\}$, $p \le n$, is calculated as

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=i_1}^{i_p} d(x_k, y_k)}, \tag{1}$$

where

(2) $$d(x_k, y_k) = \begin{cases} (x_k - y_k)^2, & \text{when attribute } k \text{ is continuous,} \\ 1, \text{ when } x_k \neq y_k & \text{and attribute } k \text{ is nominal,} \\ 0, \text{ when } x_k = y_k & \text{and attribute } k \text{ is nominal.} \end{cases}$$

In GA the survival chances of individual specimens are quantified by a fitness function. Having in mind, that the proposed algorithm should try to reduce both the size of the stored examples and the size of the attribute set without compromising the algorithm classification accuracy, the authors proposed the following fitness function:

(3) $$\text{Fitness} = \frac{1}{c_1 N_\text{R} + c_2 N_\text{E} + c_3 N_\text{A}},$$

where:

N_R is the number of training examples erroneously classified by the NN classifier using the sets of examples and attributes specified by a specimen,

N_E is the number of examples in a specimen used for classification,

N_A is the number of attributes in a specimen used for classification.

The function is controlled by three user-supplied parameters – c_1, c_2, and c_3 that weigh the user's preferences. For instance, if c_1 is high, emphasis is placed on the classification accuracy; if c_2 or c_3 are high, emphasis is placed on minimizing the number of retained examples or the number of retained attributes, respectively.

A new population is created as a result of three operations: recombination, selection and mutation. The recombination is implemented by means of a two-point crossover operator applied to a pair of probabilistically selected specimens ("parents"). The algorithm starts with selecting two pairs of integers using the uniform distribution – one pair from the closed interval [1, N_1], and another pair – from [1, N_2], where N_1 and N_2 are correspondently the chromosome length of the first and second parent. Each of the selected pairs defines a substring in the respective chromosome (the first allele and the last allele are included in the substring). The crossover operator then exchanges the substring from the first chromosome with the substring from the second chromosome. The process is applied separately to the chromosome containing the list of examples and to the chromosome containing the list of attributes. The probability of selecting a specimen S for recombination from a current Population is calculated according to the "traditional" formula:

(4) $$P(S) = \frac{\text{Fitness}(S)}{\sum_{S \in \text{Population}} \text{Fitness}(S)}.$$

The selection method determines the final structure of a new population (generation) defining the percentage of children and parents in the population. The approach proposed by Rozsypal and Kubat does not use such a parameter – instead, the percentage of the surviving parent specimens depends both on the overall quality of the previous and new generation. Let n is the size of the population. The algorithm firstly creates n children using the recombination method described above and adds all these children to their parents, creating in such a way a temporary set of specimens of size 2n. Then this set is sorted by the fitness function, and the worst n specimens are removed.

The last step in the process of creation of a new population is the mutation. The mutation operator prevents degeneration of the population's diversity and guarantees that the population represents a sufficiently large part of the search space. The authors explain the work of their mutation operator in the following way: "Our mutation operator randomly selects a prespecified percentage of the alleles in the newly created population and adds to each of them a random integer generated separately for each allele. The algorithm then takes the result modulo the number of examples/attributes. For illustration, suppose that the original number of examples/attributes is 100 and that the allele chosen for mutation contains the value 85. If the randomly generated integer is 34, then the value in the allele after mutation will be (85 + 34) mod 100 = 19".

As one can see, the cited passage needs additional interpretation in order to re-implement this mutation operator. First of all we consider that a parameter of mutation is a percentage of the *population specimens* rather than of alleles. Then for each *selected specimen* (or for each chromosome of the specimen) a random integer defining a concrete specimen (or chromosome) *allele* should be generated. Then for each *selected allele* a random integer defining new, mutated value of this allele (according to the described above original algorithm) should be selected.

The last element of a GA is the termination criterion. The authors of the original algorithm use the simple termination criterion – a fixed (user-supplied) number of generations. When the search has stopped, the specimen with the highest value of the fitness function defines both the set of the retained training examples and the set of retained attributes.

2.3. Important implementation details

An important aspect of the analyzed algorithm, in our opinion, is a method for initialization of the first population. At the beginning, all specimens have the first chromosome with the length equal to 10 and the second chromosome with the length equal to N_A – the number of attributes. All chromosomes are then filled by uniformly distributed random numbers.

The original article [17] does not explain whether the repetition of numbers is allowed in such initialization so we have assumed that the numbers are randomly generated *without* repetitions. This means that all specimens in the initial population start with the full set of attributes.

Other important details influencing on the evaluation of the algorithms efficiency are related to the experimental setting. First of all it should be taken into account that such experimental setting includes some pre-processing of data used in experiments:

1. All examples with missing attribute values are removed, and

2. All values of continuous attributes are normalized in such a way that the mean value of each attribute becomes 0 and its standard deviation becomes 1.

The second important aspect is a schema selected for evaluating the classification accuracy of the algorithm – it was used 5-fold cross-validation repeated 10 times for different partitionings of the data, and then the results were averaged.

3. Our modifications to the algorithm

Our main objectives were to analyze the original algorithm proposed in [17] and to try to increase its accuracy based on the analysis done. To achieve these objectives we have made some modifications to the algorithm and developed a special environment for the algorithm experimental evaluation.

3.1. The algorithm modifications

The first modification concerns the way for processing nominal attributes. It is a well known fact that the classification accuracy of several algorithms, which are based on calculating the similarity between objects, can be improved by application of MVDM metrics [2]. That is why, instead of formulae (1) and (2) we use:

(1')
$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=i_1}^{i_p} d^2(x_k, y_k)},$$

where

(2')
$$d(x_k, y_k) = \begin{cases} |x_k - y_k|, & \text{when attribute } k \text{ is continuous,} \\ d_{\mathrm{MVDM}}(x_k, y_k), & \text{when attribute } k \text{ is nominal,} \end{cases}$$

and

(5)
$$d_{\mathrm{MVDM}}(x_k, y_{ki}) = \frac{1}{2}\sum_{i=1}^{|C|} |P(c_i \mid x_k) - P(c_i \mid y_{ki})|, \text{ where } |C| \text{ is the number of classes}.$$

Introducing MVDM metrics has allowed us to improve also the way of processing missing attributes. In the original algorithm examples with missing attribute values were removed. In our implementation we have replaced missing values of continuous attributes with the average value of the same attribute values of all training examples belonging to the same class as the example with the missing value. For nominal attributes we have introduced a special value "?", which is interpreted as one of admissible values of such attribute and applied MVDM metrics for handling such a value.

The second modification concerns the fitness function used. We think that a number of erroneously solved examples used in the original fitness function does not allow to have an adequate impression of accuracy of the classifier since it does not take into account the size of the tested dataset. The same consideration it true for the numbers of retained attributes and training examples. Moreover, the use of these values gives no reasonable basis for selecting the "best" parameters c_1, c_2 and c_3, and may be this is the main reason, why they all were set to 1 in the original implementation.

To avoid the mentioned above deficiency we have decided to change the absolute values of N_R, N_E and N_A to its *relative*, normalized values $N^{\%}_\mathrm{R}$, $N^{\%}_\mathrm{E}$ and $N^{\%}_A$, where:

$N^{\%}_\mathrm{R}$ is calculated as a number of erroneously solve examples relative to total examples;

$N^{\%}_{\mathrm{E}}$ is a proportion of a number of retained examples among the total number of examples;

$N^{\%}_{\mathrm{A}}$ is a proportion of retained attributes relative to the total number of attributes.

The obtained fitness function looks as follows:

$$\text{Fitness} = \frac{1}{c_1 N^{\%}_{\mathrm{R}} + c_2 N^{\%}_{\mathrm{E}} + c_3 N^{\%}_{\mathrm{A}}}. \tag{3'}$$

We have also added a simple modification to the procedure of specimens' comparison – when the values of the fitness function of two specimens are equal, the priority is given to the specimen with greater accuracy (smaller value of $N^{\%}_{\mathrm{R}}$), then with greater compression rate (smaller value of $N^{\%}_{\mathrm{E}}$); and then – with greater attribute compression rate.

3.2. The experimental environment

A special experimental system – GenATE (**Gen**etic **A**lgorithm **T**esting **E**nvironment) has been developed for experiments with different kinds of genetic algorithms used for solving the classification task. The system allows making experiments on datasets represented in the traditional ".data" and ".name" text file format (Blake, Mertz, 2007). In the moment the system permits to genetically induce IF-THEN rules according to the algorithm described in [13], as well as to create the reduced nearest neighbour classifiers according to the algorithm described in [17] with our modification described above.

To facilitate the analysis of the algorithm behaviour we have decided to visualize some of the algorithm parameters – in the case of the our genetic NN algorithm they are: the greatest values of the fitness function, the classification accuracy, the example compression and attribute compression rates of the "best" specimen in the current population as well as the greatest value of the fitness function of the best specimen from all populations generated till now (Fig. 1).

For more comprehensive visualization, the fitness function is normalized:

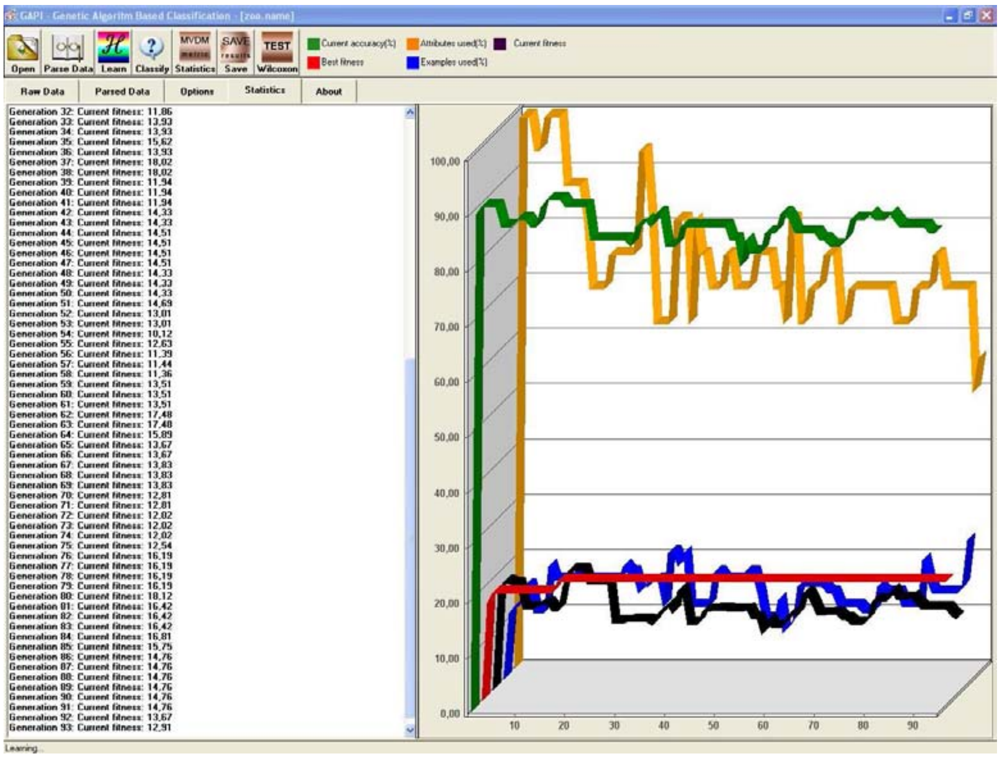
$$\text{Fitness}_{\text{Norm}} = \frac{\text{Fitness}}{\text{Fitness}_{\max}}, \tag{6}$$

where

$$\text{Fitness}_{\max} = \frac{1}{c_1 \dfrac{1}{N_{\mathrm{E}}} + c_2 \dfrac{|C|}{N_{\mathrm{E}}} + c_3 \times 1}. \tag{7}$$

In other words we consider that in order to reach the maximum of the fitness function, the "best" specimen must have at least one example per class and each example should be described at least by one attribute. In order to preserve the influence of c_1 parameter we have decided that the best accuracy is achieved when the specimen made only one error from the whole data set.

The options available for tuning by the user are split into the following groups:



```
Generation 32: Current fitness: 11,86
Generation 33: Current fitness: 13,93
Generation 34: Current fitness: 13,93
Generation 35: Current fitness: 15,62
Generation 36: Current fitness: 13,93
Generation 37: Current fitness: 18,02
Generation 38: Current fitness: 18,02
Generation 39: Current fitness: 11,94
Generation 40: Current fitness: 11,94
Generation 41: Current fitness: 11,94
Generation 42: Current fitness: 14,33
Generation 43: Current fitness: 14,33
Generation 44: Current fitness: 14,51
Generation 45: Current fitness: 14,51
Generation 46: Current fitness: 14,51
Generation 47: Current fitness: 14,51
Generation 48: Current fitness: 14,33
Generation 49: Current fitness: 14,33
Generation 50: Current fitness: 14,33
Generation 51: Current fitness: 14,69
Generation 52: Current fitness: 13,01
Generation 53: Current fitness: 13,01
Generation 54: Current fitness: 10,12
Generation 55: Current fitness: 12,63
Generation 56: Current fitness: 11,39
Generation 57: Current fitness: 11,44
Generation 58: Current fitness: 11,36
Generation 59: Current fitness: 13,51
Generation 60: Current fitness: 13,51
Generation 61: Current fitness: 13,51
Generation 62: Current fitness: 17,48
Generation 63: Current fitness: 17,48
Generation 64: Current fitness: 15,89
Generation 65: Current fitness: 13,67
Generation 66: Current fitness: 13,67
Generation 67: Current fitness: 13,83
Generation 68: Current fitness: 13,83
Generation 69: Current fitness: 13,83
Generation 70: Current fitness: 12,81
Generation 71: Current fitness: 12,81
Generation 72: Current fitness: 12,02
Generation 73: Current fitness: 12,02
Generation 74: Current fitness: 12,02
Generation 75: Current fitness: 12,54
Generation 76: Current fitness: 16,19
Generation 77: Current fitness: 16,19
Generation 78: Current fitness: 16,19
Generation 79: Current fitness: 16,19
Generation 80: Current fitness: 18,12
Generation 81: Current fitness: 16,42
Generation 82: Current fitness: 16,42
Generation 83: Current fitness: 16,42
Generation 84: Current fitness: 16,81
Generation 85: Current fitness: 15,75
Generation 86: Current fitness: 14,76
Generation 87: Current fitness: 14,76
Generation 88: Current fitness: 14,76
Generation 89: Current fitness: 14,76
Generation 90: Current fitness: 14,76
Generation 91: Current fitness: 14,76
Generation 92: Current fitness: 13,67
Generation 93: Current fitness: 12,91
```

Fig. 1. Visualization of the fitness function and some of its parameters

Basic options of a genetic algorithm:

- *Population size* – a number of specimens in a population
- *Maximum number of generations* – a maximum number of iterations the algorithm can run.
- *Crossover speed* [0, 1] – the fraction of a population to be replaced by the crossover operation at each iteration. The method for creating the next generation proposed in (R o z s y p a l, K u b a t [17]) is implemented when the value of this parameter is set to zero.
- *Mutation speed* [0, 1] – the fraction of a population to be mutated.

Options specific to the genetic Nearest Neighbour algorithm:

- *Metrics for processing nominal attributes {MVDM, Overlay}*
- c_1 – a greater value punishes a greater inaccuracy
- c_2 – a greater value punishes a greater number of examples retained
- c_3 – a greater value punishes a greater number of attributes retained

Options for algorithm evaluation methods:

- *Evaluation scheme {Hold Out, Cross-validation}*
- *Sampling method {Normal, Stratified}* – defines how each training (and testing) set will be randomly constructed. "Stratified" option preserves the original class distribution both in training and testing sets.
- *Number of Folds* (for *Cross-validation* scheme)

- *Test set size* (0, 1) – the fraction of examples used for testing when *Hold Out* evaluation scheme is used.
- *Number of Iterations* – defines how much the selected evaluation scheme should be repeated.

Options for data pre-processing:

- *Missing values {Statistic, Removal}* – defines whether the missing attribute values will be processed as it has been described in Subsection 3.1 or all examples with such values will be simply removed.
- *Discretization {yes, no}* – defines whether all continuous attributes should be discretized or not.
- *Discretization Methods {Equal width, Equal Frequency}* – defines the method for discretization.
- *Number of discretization intervals*

4. Empirical study

We carried out an extensive empirical study aimed at comparing performance of the modified algorithm with the original one and determining the role of different algorithm parameters. Sixteen benchmark datasets from widely used UCI repository [5] were used in the study. The main characteristics of these datasets are shown in Table 1.

Table 1. Main characteristics of a dataset used for algorithm evaluation

Dataset	Code	Exs.	Atts.	Cont.	Classes	Missing, %
Breast cancer	BC	286	9	0	2	0.3
Breast cancer Wisconsin	BW	699	10	10	2	2.2
Mushrooms	MU	8124	23	0	2	30.5
Pima diabetes	PI	768	8	8	2	0
Glass	GL	214	9	9	6	0
Ionosphere	IO	351	35	34	2	0
Heart diseases	HD	303	13	6	2	0.2
Balance-scale	BS	625	5	0	3	0
Hepatitis	HE	155	19	6	2	0
Iris	IR	150	4	4	3	0
New-thyroid	NT	215	6	5	3	0
Liver disease	LD	345	6	6	2	0
Tic-tac-toe	TT	958	10	0	2	0
Voting records	VO	435	16	0	2	5.6
Wine	WI	178	13	13	3	0
Zoology	ZO	101	16	1	7	0

The classification accuracy of tested algorithms was calculated by using 5-folds stratified cross-validation repeated 10 times[1] (i.e. 10×5-CV). The accuracy values on all 50 folds were used for determining the statistical significance of the results by

[1] Splitting each database in 4/5 for training and 1/5 for testing was used to allow comparison with the results reported by Rozsypal and Kubat.

means of p-values calculated according to the one-tailed t-paired test [18]. The overall behaviour of the algorithms was also evaluated by comparing their average accuracy across all databases and by means of Wilcoxon matched pair signed ranks test [18].

In order to obtain the results compatible with those reported by Rozsypal and Kubat, we fixed values of some algorithm parameters to that used in the original algorithm:

- Population size – 30 specimens;
- Number of generations – 100;
- Mutation rate – 10%;
- The crossover phase – implemented according to the original algorithm.

The first set of experiments was aimed to test our hypothesis on raising the algorithm classification accuracy by application of MVDM metrics. Six datasets containing mostly nominal attributes were selected for testing two versions of our algorithm (marked as “GenATE” in the tables) – the first one used the “traditional” overlay metrics and the second – MVDM metrics (see Table 2). In both cases the parameters of the fitness function (see eq. (3')) were the same (c_1=10, c_2=c_3 =1).

Table 2. Results of experiments with MVDM metrics: average accuracies and standard deviations; *p*-values, a number of significant wins against losses and α value of the Wilxocon test

Dataset	GenATE without MVDM metrics	GenATE with MVDM metrics	*p*-value
BC	73.25±2.62	73.47±2.61	> 0.05
MU	94.90±0.24	***98.60±0.13***	0.0005
VO	93.65±1.17	***95.57±0.99***	0.0005
BS	64.62±1.91	***82.07±1.53***	0.0005
HE	88.46±2.55	88.72±2.53	> 0.05
TT	***69.74±1.48***	69.07±1.49	0.01
ZO	88.44±3.17	89.54±3.02	> 0.05
Average	**81.87±1.88**	**85.29±1.76**	–
Sign. Wins	**1**	**3**	–
Wicoxon	–	**α = 0.05**	–

The results of these experiments have proved that the use of MVDM metrics for processing nominal attributes increases the classification accuracy of the algorithm. GenAtE algorithm with MVDM metrics has the higher average accuracy on 8 databases and as a whole is statistically better than the same algorithm with the overlay metric with 95% confidence value according to the Wilcoxon test. That is why in the rest of experiments we used MVDM instead of the overlay metrics.

The next set of experiments aimed at analyzing the behaviour of the algorithm in dependence of the fitness function used. More precisely, we tried to find “the best” values of parameter c_1, c_2, c_3 (see eq. (3')) determining the influences of the algorithm accuracy, percent of the retained examples and attributes to the overall behaviour of the algorithm. The results of these experiments are summarized in Table 3.

As it was expected, the strong emphasis on the importance of the accuracy component of the fitness function led to the statistically significant increase of the overall classification accuracy of the algorithm at the expense of an increase of the percentage of the retained examples and attributes. It has proved that our modification of the fitness function is more flexible since provides a reasonable basis for selecting values of the function parameters.

Table 3. Results of experiments with parameters of the fitness function

Dataset	$c_1 = c_2 = c_3 = 1$			c_1= 100 c_2= 10 c_3 = 1			p-value
	Accuracy	Retained examples, %	Retained attributes, %	Accuracy	Retained examples, %	Retained attributes, %	
BC	69.27±2.73	1.23±0.65	11.11±1.86	***73.29±2.61***	2.84±0.97	40.00±2.85	0.0005
BW	91.83±1.04	0.79±0.33	10.00±1.13	***96.01±0.74***	1.25±0.42	44.60±1.87	0.0005
MU	98.56±0.13	0.07±0.03	9.09±0.32	***99.03±0.1***1	0.11±0.04	32.45±0.51	0.0005
IR	93.93±1.93	4.78±1.74	25.00±3.54	94.73±1.82	8.17±2.23	56.50±3.99	> 0.05
GL	56.13±3.39	2.40±1.04	11.11±2.15	***60.69±3.33***	6.99±1.74	53.33±3.31	0.0005
WI	83.95±2.74	3.19±1.31	13.08±2.52	***94.81±1.66***	6.70±1.87	53.69±3.70	0.0005
HD	73.41±2.53	1.39±0.67	8.00±1.56	***82.85±2.16***	2.98±0.97	71.85±2.50	0.0005
VO	***95.59±0.98***	1.26±0.53	12.75±1.60	95.29±1.02	1.69±0.61	25.38±2.08	0.025
PI	73.52±1.59	0.51±0.26	12.50±1.19	***74.52±1.57***	0.88±0.34	29.25±1.63	0.01
IO	81.83±2.05	1.20±0.58	3.76±1.01	***86.35±1.83***	4.07±1.05	24.88±2.25	0.0005
BS	59.83±1.96	0.67±0.33	25.00±1.73	***82.25±1.53***	2.03±0.56	100.00±0.00	0.0005
HE	88.27±2.57	2.82±1.32	5.58±1.84	88.02±2.60	5.82±1.87	24.63±3.41	> 0.05
LD	61.10±2.62	1.02±0.54	16.67±2.01	***64.00±2.58***	2.60±0.85	40.00±2.62	0.01
NT	90.47±2.00	2.70±1.10	20.00±2.73	***93.30±1.70***	5.98±1.61	59.60±3.30	0.0005
TT	65.12±1.54	0.26±0.16	11.11±1.02	***72.66±1.44***	1.63±0.41	86.44±0.99	0.0005
ZO	58.72±4.89	6.29±2.41	13.63±3.39	***90.23±2.94***	18.72±3.87	80.75±3.81	0.0005
Average	**77.59±2.17**	**1.91±0.81**	**13.02±1.85**	**84.25±1.85**	**4.53±1.21**	**51.46±2.43**	–
Sign. wins	**1**	–	–	**13**	–	–	–
Wicoxon	–	–	–	**α = 0.05**	–	–	–

Table 4. Results of experiments with parameters of the fitness function

Dataset	c_1= 100 c_2= 10 c_3 = 1			c_1= 10 c_2= 0 c_3 = 0			p-value
	Accuracy	Retained examples, %	Retained attributes, %	Accuracy	Retained examples, %	Retained attributes, %	
BC	73.29±2.61	2.84±0.97	40.00±2.85	73.36±2.61	3.21±1.04	50.22±2.85	> 0.05
BW	96.01±0.74	1.25±0.42	44.60±1.87	96.21±0.72	1.33±0.43	67.00±1.76	> 0.05
MU	99.03±0.11	0.11±0.04	32.45±0.51	***99.43±0.08***	0.16±0.04	71.91±0.49	0.0005
IR	94.73±1.82	8.17±2.23	56.50±3.99	***95.53±1.67***	8.73±2.30	71.00±3.65	0.05
GL	60.69±3.33	6.99±1.74	53.33±3.31	59.31±3.35	6.74±1.71	60.44±3.27	> 0.05
WI	94.81±1.66	6.70±1.87	53.69±3.70	94.59±1.68	7.66±1.99	59.69±3.54	> 0.05
HD	82.85±2.16	2.98±0.97	71.85±2.50	82.75±2.17	2.84±0.95	72.15±2.51	> 0.05
VO	95.29±1.02	1.69±0.61	25.38±2.08	95.31±1.01	1.94±0.66	34.38±2.24	> 0.05
PI	74.52±1.57	0.88±0.34	29.25±1.63	74.51±1.57	0.95±0.34	35.00±1.69	> 0.05
IO	86.35±1.83	4.07±1.05	24.88±2.25	86.19±1.84	4.47±1.10	32.47±2.48	> 0.05
BS	82.25±1.53	2.03±0.56	100.00±0.00	81.96±1.54	2.21±0.59	100.00±0.00	> 0.05
HE	88.02±2.60	5.82±1.87	24.63±3.41	87.75±2.62	6.75±1.99	27.58±3.58	> 0.05
LD	64.00±2.58	2.60±0.85	40.00±2.62	64.17±2.57	2.71±0.87	41.00±2.62	> 0.05
NT	93.30±1.70	5.98±1.61	59.60±3.30	93.44±1.68	6.13±1.62	66.40±3.21	> 0.05
TT	72.66±1.44	1.63±0.41	86.44±0.99	72.44±1.44	1.78±0.43	87.11±1.04	> 0.05
ZO	90.23±2.94	18.72±3.87	80.75±3.81	90.09±2.95	19.74±3.96	81.63±3.79	> 0.05
Average	**84.25±1.85**	**4.53±1.21**	**51.46±2.43**	**84.19±1.84**	**4.83±1.25**	**59.87±2.42**	–
Sign. wins	**0**	–	–	**2**	–	–	–
Wicoxon	–	–	–	**α > 0.05**	–	–	–

In most cases, the main objective of selecting the most representative training examples and relevant attributes is to raise the classification accuracy of the NN-algorithms. Therefore, the presence of such unrepresentative examples or examples described by irrelevant attributes will automatically reduce the classification accuracy of a specimen constructed from such examples. In other words, this *implicit* dependence between specimen classification accuracy and the quality of examples representing the specimen may be simply modelled by a fitness function, which penalizes only the loss of specimen accuracy. In our case, such fitness function can be easy achieved by setting parameters $c_2=c_3=0$. Table 4 presents results of experiments with such fitness function. It should be mentioned, that we still preserve an evolutionary way for selecting representative examples and attributes in each specimen of the population, but evaluate the quality of such specimens only based on their classification accuracy.The results show that a more simple fitness function leads to practically the same average classification accuracy of the algorithm. Although this increase is not statistically significant from the Wilcoxon test point of view, we can see that in 2 databases the tested fitness function has achieved statistically significant increase in the classification accuracy. It is interesting to see that the better classification accuracy has been achieved with practically the same percent of the retained examples and comparatively low increase of the retained attributes.

In the next experiments we tested the dependency of the classification accuracy of the algorithm from the method used for selecting specimens for new generation. In the approach proposed by Rozsypal and Kubat the percentage of the surviving parent specimens depends both on the overall quality of the previous generation and on the quality of the new specimens – children (see Section 2). We have compared this method with another popular selection method [13], which determines the final structure of a new generation by defining a fixed percentage of children and parents in it. Table 5 presents the results of such a comparison. Both versions of the algorithm were tested by using the same fitness function with parameters $c_1 = 100$, $c_2 = 10$ and $c_3 = 1$. In the second selection method 60% of specimens[2] from the old generation were randomly selected (based on their values of the fitness function) to become members of the next generation, and the rest 40% of the new generation was populated by children of specimens from the previous generation.

The results have shown that this “traditional” method for constructing new generations (with a fixed proportion of parents and children) is significantly better from Wilcoxon test point of view than the method used by Rozsypal and Kubat. It is worth to be mentioned that the average increase of classification accuracy is accompanied by a slight decrease both in the number of retained examples and attributes.

Table 6 shows the classification accuracies of our algorithm (GenATE), the original algorithm (GA-RK) and the Nearest Neighbour algorithm (1-NN). Both GenATE and 1-NN used the same distance function (Euclidian – for processing continuous attributes and MVDM – for nominal attributes) and were tested on the same folds in the GenATE environment. GenATE algorithm used the method for creating new generations according to that described in [13] and parameters of the fitness function were $c_1 = 10$, $c_2 = 0$ and $c_3 = 0$.

[2] These are the most frequently used values for such parameters.

Table 5. Results of experiments comparing different methods for selecting specimens for new generations

Dataset	Selection based on [17]			Selection based on [13]			*p*-value
	Accuracy	Retained examples, %	Retained attributes, %	Accuracy	Retained examples, %	Retained attributes, %	
BC	73.29±2.61	2.84±0.97	40.00±2.85	73.68±2.60	2.85±0.98	43.33±2.90	> 0.05
BW	96.01±0.74	1.25±0.42	44.60±1.87	95.84±0.75	1.36±0.44	46.20±1.88	> 0.05
MU	99.03±0.11	0.11±0.04	32.45±0.51	***99.23±0.10***	0.13±0.04	36.27±0.52	0.01
IR	94.73±1.82	8.17±2.23	56.50±3.99	94.47±1.86	8.08±2.22	59.50±4.00	> 0.05
GL	60.69±3.33	6.99±1.74	53.33±3.31	61.30±3.32	6.19±1.64	49.11±3.35	> 0.05
WI	94.81±1.66	6.70±1.87	53.69±3.70	95.12±1.59	6.07±1.79	45.69±3.71	> 0.05
HD	82.85±2.16	2.98±0.97	71.85±2.50	82.12±2.20	2.90±0.96	65.38±2.65	> 0.05
VO	95.29±1.02	1.69±0.61	25.38±2.08	95.47±1.00	1.87±0.65	26.75±2.11	> 0.05
PI	74.52±1.57	0.88±0.34	29.25±1.63	74.94±1.56	0.84±0.33	27.50±1.60	> 0.05
IO	86.35±1.83	4.07±1.05	24.88±2.25	86.64±1.81	3.95±1.03	18.53±2.04	> 0.05
BS	82.25±1.53	2.03±0.56	100.00±0.00	82.72±1.51	1.98±0.55	100.00±0.00	> 0.05
HE	88.02±2.60	5.82±1.87	24.63±3.41	88.31±2.58	5.22±1.78	19.37±3.07	> 0.05
LD	64.00±2.58	2.60±0.85	40.00±2.62	64.41±2.58	2.59±0.84	39.00±2.62	> 0.05
NT	93.30±1.70	5.98±1.61	59.60±3.30	93.77±1.63	5.58±1.56	56.40±3.37	> 0.05
TT	72.66±1.44	1.63±0.41	86.44±0.99	72.30±1.45	1.72±0.42	79.78±1.27	> 0.05
ZO	90.23±2.94	18.72±3.87	80.75±3.81	***92.01±2.69***	19.63±3.95	75.13±4.20	0.025
Average	**84.25±1.85**	**4.53±1.21**	**51.46±2.43**	**84.52±1.83**	**4.44±1.20**	**49.25±2.45**	–
Sign. Wins	**0**	–	–	**2**	–	–	–
Wicoxon	–	–	–	**α = 0.05**	–	–	–

Table 6. Classification accuracies of GA-RK, GenATE and 1-NN algorithms

Dataset	GA-RK	GenATE ($c_1 = 10$, $c_2 = 0$, $c_3 = 0$)			1-NN	*p*-value
		Accuracy	Retained examples, %	Retained attributes, %		
BC	N/A	***73.18±2.62***	3.15±1.03	50.44±2.93	69.52±2.72	0.0005
BW	95.9±1.2	***95.88±0.75***	1.25±0.42	60.20±1.84	95.48±0.79	0.05
MU	99.4±0.4	99.52±0.08	0.13±0.04	60.36±0.53	***100.00±0.00***	0.0005
IR	94.5±2.3	94.53±1.85	9.47±2.38	67.50±3.80	95.27±1.73	> 0.05
GL	57.3±5.4	60.70±3.33	6.67±1.70	66.67±3.16	***69.07±3.16***	0.0005
WI	90.4±3.1	95.37±1.55	7.08±1.92	57.54±3.67	95.01±1.63	> 0.05
HD	77.9±5.4	***82.19±2.19***	2.88±0.96	67.08±2.67	79.28±2.33	0.0005
VO	61.9±5.7	***95.24±1.02***	2.19±0.70	40.25±2.31	94.07±1.13	0.0005
PI	74.9±2.6	***74.73±1.57***	1.00±0.36	30.00±1.64	70.75±1.64	0.0005
IO	84.0±4.6	***87.47±1.77***	4.52±1.10	23.71±2.24	86.44±1.83	0.05
BS	N/A	83.00±1.50	1.88±0.54	100.00±0.00	***84.18±1.46***	0.05
HE	N/A	87.03±2.69	5.90±1.89	26.21±3.50	87.63±2.64	> 0.05
LD	N/A	***64.03±2.58***	2.73±0.87	41.00±2.61	61.54±2.62	0.025
NT	N/A	94.42±1.55	5.56±1.56	58.00±3.33	***96.33±1.27***	0.0005
TT	N/A	72.06±1.45	1.69±0.42	83.78±1.16	***93.80±0.78***	0.0005
ZO	N/A	91.06±2.81	20.73±4.02	75.00±4.29	92.34±2.65	> 0.05
Average	–	**84.40±1.83**	**4.80±1.24**	**56.73±2.48**	**85.67±1.77**	–
Sign. wins	–	**7**	–	–	**5**	–
Wicoxon	–	**α > 0.05**	–	–	–	–

The values of GA-RK accuracies have been taken from [17]. Unfortunately these accuracies can not be used directly for comparison with the results of our algorithm since they were achieved on the datasets described by artificially extended sets of attributes (see [17] for details).

The results have shown that GenATE algorithm is able to achieve practically the same classification accuracy (in average) as the nearest neighbour classifier, which is known as one of the most accurate classifier. Moreover, this very high classification accuracy is achieved by using, in average, less than 5% of training examples and less than 60% of attributes.

In the last set of experiments we tried to understand the roles of explicit (evolutionary) and implicit (initialization) parts of the tested genetic algorithm. In order to do this we have measured the classification accuracy of the first population[3] randomly created in accordance with the original algorithm initialization procedure (see Section 2.3) and compared it with the final accuracy of the algorithm achieved after applying the described above evolutionary search method. For these experiments we have slightly changed the original initialization procedure by applying *a stratified* random method for selecting examples, which guarantees that each specimen in the initial population will contain at least one training example per class. Table 7 shows the accuracy of such "only-one-population" method in comparison with the results achieved after completing the evolutionary search. The last column shows the increase in the algorithm accuracy caused by applying the evolutionary approach. The parameters of the fitness function were $c_1 = 10$, $c_2 = 0$ and $c_3 = 0$.

Table 7. Results of experiments evaluating the role of the evolutionary mechanism used in the tested algorithm

Data-set	1 generation			100 generations			*p*-value	Acc. Δ(%)
	Accuracy	Retained examples, %	Retained attributes, %	Accuracy	Retained examples, %	Retained attributes, %		
BC	70.07±2.71	4.37±1.21	100.00	***73.18±2.62***	3.15±1.03	50.44±2.93	0.0005	4.44
BW	95.88±0.75	1.79±0.50	100.00	***96.47±0.70***	1.25±0.42	60.20±1.84	0.005	0.61
MU	98.99±0.11	0.15±0.04	100.00	***99.52±0.08***	0.13±0.04	60.36±0.53	0.0005	0.53
IR	94.53±1.85	8.33±2.26	100.00	95.60±1.65	9.47±2.38	67.50±3.80	> 0.05	1.12
GL	52.59±3.41	5.84±1.60	100.00	***60.70±3.33***	6.67±1.70	66.67±3.16	0.0005	15.43
WI	93.72±1.79	7.02±1.92	100.00	***95.37±1.55***	7.08±1.92	57.54±3.67	0.025	1.76
HD	82.19±2.19	4.13±1.14	100.00	82.58±2.18	2.88±0.96	67.08±2.67	> 0.05	0.47
VO	94.71±1.07	2.87±0.80	100.00	***95.24±1.02***	2.19±0.70	40.25±2.31	0.025	0.56
PI	71.14±1.63	1.63±0.46	100.00	***74.73±1.57***	1.00±0.36	30.00±1.64	0.0005	5.04
IO	84.08±1.95	3.56±0.99	100.00	***87.47±1.77***	4.52±1.10	23.71±2.24	0.0005	4.03
BS	81.24±1.56	2.00±0.56	100.00	***83.00±1.50***	1.88±0.54	100.00	0.005	2.18
HE	82.82±3.02	8.06±2.19	100.00	***87.03±2.69***	5.90±1.89	26.21±3.50	0.0005	5.08
LD	59.19±2.64	3.62±1.01	100.00	***64.03±2.58***	2.73±0.87	41.00±2.61	0.0005	8.18
NT	92.51±1.79	5.81±1.60	100.00	***94.42±1.55***	5.56±1.56	58.00±3.33	0.001	2.06
TT	71.66±1.46	1.30±0.37	100.00	72.06±1.45	1.69±0.42	83.78±1.16	> 0.05	0.56
ZO	91.06±2.81	12.38±3.28	100.00	91.31±2.78	20.73±4.02	75.00±4.29	> 0.05	0.27
Ave-rage	**82. 27±1.92**	**4.56±1.24**	**100.00**	**84.54±1.81**	**4.80±1.24**	**56.73±2.48**	**–**	**2.76**
Sign. wins	**0**	**–**	**–**	**12**	**–**	**–**	**–**	**–**
Wilco-xon	**–**	**–**	**–**	**$\alpha = 0.05$**	**–**	**–**	**–**	**–**

The results of these experiments are very interesting – although the use of the evolutionary techniques has led to creating the statistically more accurate classifier, the average impact of these techniques is less than 3% in average on 16 benchmark datasets! Thus the rest 97% of the overall classification accuracy of the classifier is due to the *implicit mechanism* used for its initialization.

In other words, a rather accurate edited nearest neighbour classifier (we will call it "randomized nearest neighbour") can be created by means of a very simple procedure:

[3] This value is calculated as the accuracy of the "best" (i.e. with the highest value of fitness function) specimen in the initial population.

Step 1. Randomly select a small set of training examples with size $k<<n$ including at least one training example per class (n is the number of all training examples)

Step 2. Repeat the previous step $m << n$ times and create m candidate classifiers.

Step 3. Test m candidate classifiers on the whole training dataset.

Step 4. Select a classifier with the highest training accuracy as the final edited nearest neighbour classifier.

Having in mind, that the genetic algorithms implement a randomized beam search the experiments have shown that the initial width of the beam is more important (in our case) than the evolutionary strategies used for the search.

It is also interesting to see what are the main resources used by the tested genetic algorithm for improving the classification accuracy of the randomized nearest neighbour classifier. Table 7 has shown that such an improvement is achieved mainly by removing irrelevant attributes – the final classifier uses (in average) less than 60% of all attributes while the number of the retained examples remains practically the same.

5. Conclusions and future trends

In this paper we have considered the applicability of genetic algorithms for the problem of selecting representative examples and attributes for edited nearest neighbour classifiers. We have analysed in details one of the best such algorithms reported in the literature - that proposed by Rozsypal, Kubat, 2003. We have empirically proved that this algorithm can be improved by using MVDM metrics, modifying the fitness function and changing the method for creating new generations.

We have confirmed that the application of genetic algorithms allows creating very effective edited nearest neighbour classifiers which classification accuracies are comparable with the accuracy of the classical nearest neighbour algorithm using for classification the whole training set of examples. Based on experiments with 16 benchmark datasets we have shown that such edited nearest neighbour classifiers use, in average, less that 5% of training examples and less than 60% of attributes.

Our analysis of the algorithms constructed based on the approach proposed by Rozsypal and Kubat has also shown that although the application of genetic algorithms allows creating more accurate edited nearest neighbour classifiers, the average impact of such an application is less than 3% in average measured on 16 benchmark datasets. Thus the about 97% of the overall classification accuracy of such a classifier can be explained by the implicit non-evolutionary mechanism used for the algorithm initialization.

We are going to continue our research in several directions – first of all we plan to test our modified algorithm on a more wide range of datasets. Since the benchmark datasets used in the experiments are known to have comparatively small number of irrelevant attributes, we are going to repeat the experiments described by Rozsypal and Kubat with datasets described by artificially extended sets of irrelevant attributes.

We are also going to continue experiments with the randomized nearest neighbour algorithm in order to understand its potential and limitations.

R e f e r e n c e s

1. A g r e, G. An Integrated Prototype-Based Learning Algorithm. – Cybernetics and Information Technologies, Vol. **1**, 2001, No1, Sofia, Bulgarian Academy of Sciences, 56-70.
2. A g r e, G., S. P e e v. On Supervised and Unsupervised Discretisation. – Cybernetics and Information Technologies, Vol. **2**, 2002, No 2, Sofia, Bulgarian Academy of Sciences, 43-57.
3. A h a, D. A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations. Doctoral Dissertation, Department of Information and Computer Sciences, University of California, Irvine, 1990.
4. Al m u a l l i m, H., T. D i e t t e r i c h. Learning Boolean Concepts in the Presence of Many Irrelevant Features. – Artificial Intelligence, **69 (1-2)**, 1994, 279-306.
5. A s u n c i o n, A., D. J. N e w m a n. UCI Machine Learning Repository. Irvine, CA: University of California, Department of Information and Computer Science, 2007. **http://www.ics.uci.edu/~mlearn/MLRepository.html**
6. C a r d i e, C. Using Decision Trees to Improve Case-Based Learning. – In: Proc. Tenth Intern. Conference on Machine Learning, Morgan Kaufmann Publ., 1993, 25-32.
7. C o v e r, T., P. H a r t. Nearest Neighbour Pattern Classification. – IEEE Transactions on Information Theory, **IT-13**, 1967, 21-27.
8. F r i e d m a n, J., J. B e n t l e y, R. F i n k e l. An Algorithm For Finding Best Matches in Logarithmic Expected Time. – Transaction on Mathematical Software, Vol. **3**, 1977, No 3.
9. K o h a v i, R. Wrappers for Performance Enhancement and Oblivious Decision Graphs. Doctoral Dissertation, Department of Computer Sciences, Stanford University, 1995.
10. K o n o n e n k o, M. R o b n i k-S i k o n j a, U. P o m p e. Relief for Estimation and Discretization of Attributes in Classification, Regression, and ILP Problems. – In: Artificial Intelligence: Methodology, Systems, Applications. Ramsay (Ed.). IOS Press, 1996, 31-40.
11. K u n c h e v a, L., L. C. J a i n. Nearest-Neighbor Classifier: Simultaneous Editing and Feature Selection. – Pattern Recognition Letters, **20**, 1999, 1149-1156.
12. M c K e n n a, E., B. S m y t h. Competence-Guided Case-Base Editing Techniques. – In: Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000, 186-197.
13. M i t c h e l l, T. M. Machine Learning. The McGraw-Hill Companies, Inc., 1997.
14. M u r e s a n, D. A. Genetic Algorithms for Nearest Neighbor.1997.
15. **www.cs.caltech.edu/~muresan/GANN/report.html**
16. R o z s y p a l, A., M. K u b a t. Using the Genetic Algorithm to Reduce the Size of a Nearest-Neighbor Classifier and to Select Relevant Attributes. – In: Proc. Eighteenth International Conference on Machine Learning, Williamstown, Massachusetts, 2001, 449-456.
17. R o z s y p a l, A., M. K u b a t. Selecting Representative Examples and Attributes by a Genetic Algorithm. – Intelligent Data Analysis **7(4)**, 2003, 291-304.
18. S i n c h i c h, T. Statistics by Examples. Dellen Publishing Comp., 1990.
19. Z a n i o l o, C., S. C e r i, C h. F a l o u t s o s, R. S n o d r g r a s s, V. S u b r a h m a n i a n, R. Z i c a r i. Advanced Database Systems. Morgan Kaufmann Publ., 1997.