USING GENETIC ALGORITHM TO OPTIMIZE WEIGHTS IN DATA MINING TASK

Irina Provorova, Serge Parshutin, Sergejs Provorovs

Keywords: Genetic Algorithm, Genetic Operators Modification, Genetic Operators Control, K-Nearest Neighbour Algorithm, Data Mining Task

This paper considers an application of genetic algorithm (GA) to optimize weights in data mining task. Data mining tasks usually have datasets containing a large number of records and features that will be processed using, for example, created classification rules. As a result, by using classical method to classify a large number of records and features, a high classification error value will be obtained. To solve this problem, the genetic algorithm was applied to find for each feature the weight that would reduce classification error value.

As a classical method, the k-nearest neighbour (KNN) classifier was chosen and the modified genetic algorithm was applied to optimize the weight. Based on the joint application of genetic and k-nearest neighbour algorithms, the GA/KNN hybrid algorithm was developed. As a result, the developed hybrid algorithm provides a stable classification error reducing regardless of the number of records and features, and also of the chosen number of neighbours. In the GA block the modified crossover and mutation works in each generation with identical intensity and cannot provide debasing of the individual.

1. Introduction

This paper addresses genetic algorithm application aimed to improve the classification result. Most applications of genetic algorithm (GA) in data mining tasks optimize some parameters in the classification process. The process of finding an optimal algorithm and its control parameters for building a predictive model is non-trivial because of two reasons. First, the number of classification algorithms and their control parameters are very large. Second, it can be quite time consuming to build a model for datasets containing a large number of records and features. These two reasons make it impractical to enumerate through every algorithm and its possible control parameters for finding an optimal model.

The paper presents the application of real parameter genetic algorithm [9] in finding optimal set of weights for feature distance in the k-nearest neighbour algorithm (KNN) that reduces classification error. The obtained weight set is applied to influence on the importance of the distance between each feature value from training set and query instance corresponding feature value. As a result of the joint application of two algorithms, the GA/KNN hybrid algorithm was developed.

2. K-Nearest Neighbour Algorithm

The k-nearest neighbour algorithm (KNN) is a supervised learning algorithm where the result of new instance query is classified based on the majority of k-nearest neighbour category [2].

The data of the KNN can be any measurement scale from ordinal, nominal, to quantitative scale. The purpose of this algorithm is to classify a new object based on training and testing samples of instances. The KNN classifier does not create any model - it uses the minimum distance from the query instance to each training instance to determine the k-nearest neighbours that classify the query instance.

3. Genetic Algorithm

A genetic algorithm is based on evolutionary process in nature and uses similar terminology [1, 8]. The algorithm evaluates a finite set or "population" of individuals, based on the process of evolution (selection, crossover and mutation are applied as operators of GA). The operators have many modifications; they are performed in cycles and called generations. In this paper two types of crossover (double inversion and restricted) and mutation (augmentation and restricted) operators are applied.

3.1. Double Inversion Crossover

Double inversion crossover performs in such a way that in each selected individual couple all parts are mixed [4]:

- randomly take a pair of two individuals;
- randomly choose the crossover point;
- gene values before crossover point from the first individual replace gene values after crossover point. The replaced gene values are placed in the second individual gene values before the crossover point (the recombination of gene values in the second individual will be done analogically).

Double inversion crossover provides the increased recombination effect in the offspring due to the exchange of individuals x and y parts. As a result of the aforementioned, double inversion crossover provides higher difference in the obtained offsprings.

3.2. Restricted Crossover

The basic operator for producing new individuals in GA is crossover. Sometimes the task has a particular point – the sum of individual values would not change or would be equal to some value permanently. Usually after individuals crossing, this sum may be changed and, to keep the sum of weight values, the crossover operator has the following modifications [3]:

- randomly take a pair of two individuals;
- randomly choose the crossover point;
- calculate a sum of weight variables from the individuals that are located before and after crossover point;
- shift vice verse the respective parts of the pair of individuals in the rough guide of the crossover point;
- compare a sum of weight values that are located in the shifted part from the other individual before and after crossover. In case if these sums are different, the part which belongs to the individual with the worst fitness will have some correction (after crossover the difference value will be subtracted or added to the part of weakness individual that is located before/after crossover point).

3.3. Augmentation Mutation

Mutation operator has a low rate in comparison with the crossover rate and may cause a finding or a loss of a very good solution. To solve this problem, the mutation operator was modified in such a way to help to generate only a good solution and to review much more than only one mutation. The new individual creation process, applying augmentation mutation, is executed as follows [4, 5]:

- randomly choose an individual;
- randomly choose different augmentation for gene from the defined domain;
- calculate all possible combinations using selected individual values and the augmentation Δ;
- calculate the fitness value for each obtained individual;
- compare fitness function values and choose the fittest individual;
- replace the selected individual with the fittest individual. In case if all obtained individuals are weaker than the selected individual, the replacement will never be used. The next mutation occurs in the obtained individual. This operator will mutate until the obtained individual fitness does not cause any loss of fitness.

3.4. Restricted Mutation

To review more than just one mutation, to generate only good solution and to satisfy some conditions like, the sum of individual values would not be changed or would be equal to some value permanently [3].

The new individual creation process applying the modified mutation has the following steps:

- randomly choose an individual;
- randomly choose some weight values from the individual. The number of chosen weight values is generated by random;
- calculate the sum of all chosen weight values from the individual and then randomly generate the same number to the chosen weight values. The sum of generated weighs will not be changed;
- calculate the fitness value for the obtained individual;
- compare the chosen and the obtained individuals fitness function values;
- replace the selected individual with the fittest individual. In case if the obtained individual is fitter than the chosen one, mutation will be continued. The next mutation occurs in the obtained individual. This operator will mutate until the obtained individual fitness does not cause a loss of fitness.

4. Application of Crossover and Mutation Operators Control for Obtaining Offspring

To provide control of crossover and mutation operators, it was decided to keep worse individual fitness value in the Depository of Weak Individuals Fitness Values (DWIFV) during all generations and any generation of the same individuals' week fitness will be restricted – the fittest one will replace it (see Figure 1 and Figure 2). Hence, the domain of possible solutions becomes progressively smaller and the possibility of generating the same weak individual is protected. The search space becomes smaller in each next population that makes the genetic algorithm convergence much easier [4, 5, 6].

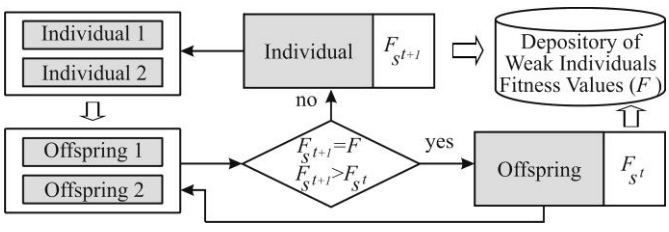

Fig. 1. Offspring Generation Control in Crossover

As Figure 1 shows, the comparison of individuals and offspring fitness values during crossover operator run is executed as follows:

- compare the chosen individuals fitness values (F_{s^t}) and those kept in DWIFV (*F)*;
- compare the obtained offspring fitness ($F_{s^{t+1}}$);
- compare the better individual and offspring fitness values F_{s^t} and $F_{s^{t+1}}$, and worse individual and offspring fitness values accordingly. After comparison of each pair the worst individual fitness value will be kept in the DWIFV during all generations.

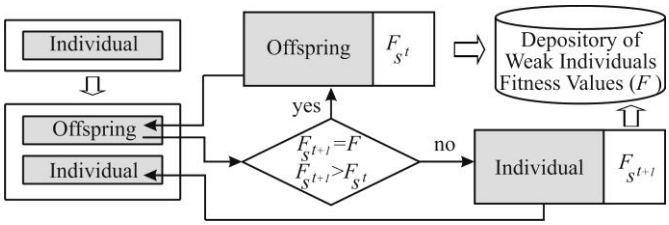

Fig. 2. Offspring Generation Control in Mutation

The mutation operator in this paper consists of several steps – individual's mutation will be repeated until better offspring is found. If better solution is found, then the individual selected for mutation will be kept during all generations in the DWIFV. The controlled mutation operator steps are the following:

- compare the chosen individuals fitness values (F_{s^t}) and kept in DWIFV (*F*);
- compare the individual (*F*) and its offspring fitness function values ($F_{s^{t+1}}$) and choose the fittest one and replace the selected individual with the fittest offspring, and "selected" individual check in the DWIFV. Repeat mutation while the obtained offspring does not get weaker.

The control of crossover and mutation operators uses different approaches in Figure 1 and Figure 2. Each pair of individuals was crossed only one time, but during each individual mutation more than one mutation step was done while the obtained offspring does not get weaker.

5. GA/KNN Hybrid Algorithm

The GA/KNN hybrid algorithm is based on two algorithm where each of them has its own specific aim - the k-nearest neighbour algorithm classify instance using the distances between them and the genetic algorithm searches for the "best" weight *w* set that helps to reduce the classification error. Namely, the weight set is applied to increase importance or decrease importance of the distance between each feature value from training set and query instance corresponding feature value. The whole structure of GA/KNN hybrid algorithm is shown in Figure 3.

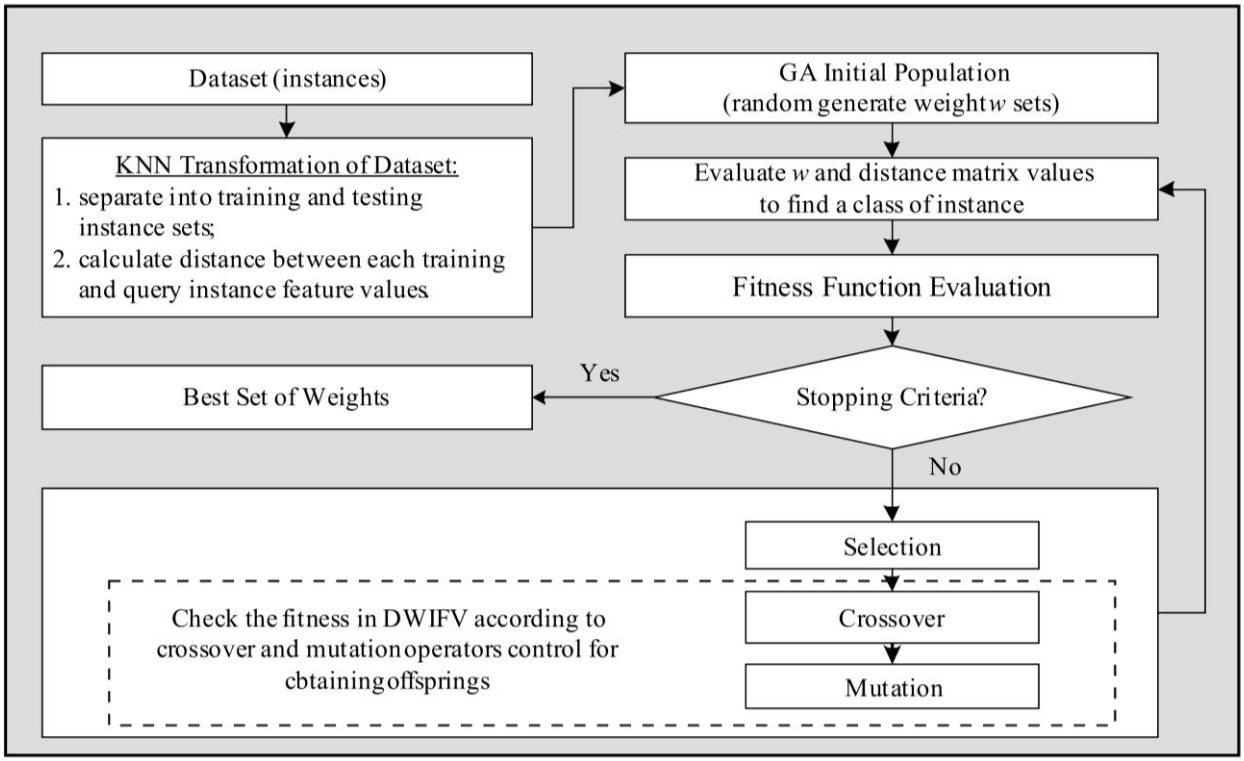

Fig. 3. Structure of GA/KNN Hybrid Algorithm

The GA/KNN hybrid algorithm works with statistical dataset and generated population values that will be evaluated together. The KNN requires that dataset values should be of continuous data type and normalized. In case if the data type is mixed, the dataset values will be processed to one data type or other data types would not be used. The GA initial population is generated randomly and consists of a number of individuals that are represented as real parameter values called by weights. The individual length is equal to the number of statistical dataset instance feature. Fitness function is calculated as follows:

$$F = \frac{Total\ Test - Incorrect}{Total\ Test}, \qquad (1)$$

where

Total Test – the number of instances of testing set;

Incorrect – the number of instances that were incorrectly classified.

The result of GA/KNN hybrid algorithm is the weight set. As weight generation, two approaches were used:

- each weight value *w* is real parameter from [0; 1];
- the sum of all weight values *w* that is real parameter from [0; 1] equals to 1.The obtained weight values will change the effect of distance between each instance feature value from training set and corresponding instance feature value that will reduce classification error of KNN whose efficiency subjected to the successful k-neighbour number choice and dataset instance values.

6. Experiments

The aim of the experiments is to prove that GA can be useful to optimize the classification result of KNN algorithm that cannot provide stable high classification accuracy. The experiments were made using the German credit dataset [7]. The number of features is 20 (7 numerical and 13 categorical).

The data is represented as numerical and categorical types and also as edited with several indicator variables that are added to make it suitable for algorithms which cannot cope with categorical variables for algorithms that need numerical features. Several features that are ordered categorical (such features as Job - unemployed/ unskilled - non-resident, unskilled – resident, skilled employee/ official, management/ self-employed/ highly qualified employee/ officer) have been coded as integer.

Each experiment group consists of 20 runs aimed to determine the average algorithmic results of weight *w* set finding that helps to reduce the classification error. The estimation of GA/KNN hybrid algorithm implementation is based on comparison with the KNN classification result, namely - the classification error. To solve the described task, two groups of experiments were performed (see Table 1).

Table 1

The parameters of experiments

Parameter name	Experiment group 1	Experiment group 2
Population size	50	50
Generation No.	100	100
Crossover rate	0.7	0.7
Mutation rate	0.1	0.1
Selection type	Tournament	Tournament
Crossover type	Double Inversion	Restricted
Mutation type	Augmentation	Restricted

7. Experimental Results

The results of all experiments produced by the GA/KNN hybrid algorithm were evaluated together. The average result values of 20 experiments for each experimental group are described in Table 2, Table 3 and Table 4 correspondingly. Also, these tables represent the estimation of classification error (%) of KNN and GA/KNN hybrid algorithms application using the same k-neighbour and instance number in the experiments.

Table 2

60 instance classification error, %

Number of K	KNN	GA/KNN, group 1	GA/KNN, group 2
k=3	18.33	4.19	7.76
k=5	28.33	6.28	8.09
k=7	25.00	6.44	5.93

The obtained results in Table 2 show that the GA/KNN hybrid can make the classification error approximately three times less.

Table 3

100 instance classification error, %

Number of K	KNN	GA/KNN, group 1	GA/KNN, group 2
k=3	22.00	11.42	10.91
k=5	24.00	13.56	10.02
k=7	23.00	11.87	13.11
k=10	22.00	11.29	10.76

The number of classified instance in Table 3 has grown up almost twice in comparison with the instance

number in Table 2, but the reduction in classification error provided by GA/KNN hybrid algorithm is very impressive - it can reduce the classification error approximately twice.

Table 4

150 instance classification error, %

Number of K	KNN	GA/KNN, group 1	GA/KNN, group 2
k=3	26.67	17.58	18.47
k=5	30.00	19.00	17.78
k=7	28.00	18.73	19.32
k=10	24.67	20.14	19.09
k=12	22.67	18.37	19.96

The experimental results of Table 4 show that GA/KNN hybrid algorithm keeps the average classification error reduction about 7.5%. All experimental results represented in the above tables are shown graphically in Figure 4.

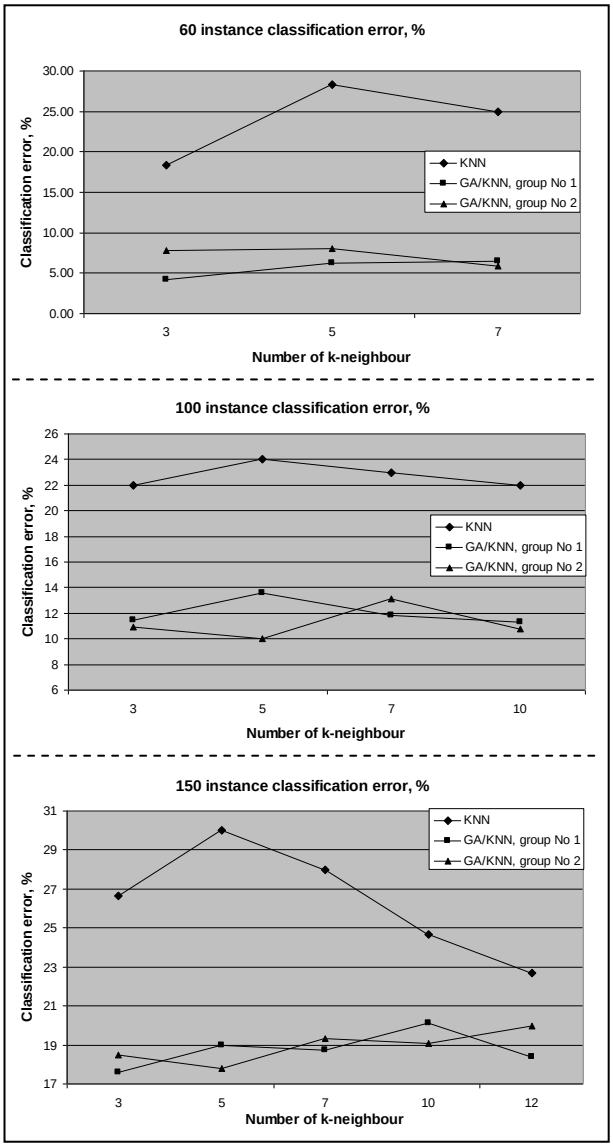

Fig.4. KNN and GA/KNN classification errors, %

As the values of classification error show, the result of GA/KNN application is much more efficient than that of the simple KNN application (see Figure 4). And if the influence of choice of the data set instance and feature number in KNN algorithm on classification error is high (the larger the data set instance and feature number, the higher classification error). Also the choice of k-neighbour number affects the KNN algorithm result (this parameter does not have proportional influence; it is based only on successful k-neighbour choice). Hence, the application of GA/KNN provides a stable classification error reduction apart from the number of records and features, and also from k-neighbour choice.

The results obtained demonstrate that the GA/KNN hybrid algorithm provides better results in credit dataset classification task than the simple KNN algorithm. It may extend because each weight value corrects the significance of distance between each instance feature value from training set and corresponding instance feature value from the testing set. Also it may extend because the modified crossover and mutation works in each generation with identical intensity and cannot provide debasing of the individual (set of weights *w*).

8. Conclusions

The GA/KNN hybrid algorithm provides much better classification results than simple KNN algorithm – the classification error is visibly reduced. The GA/KNN hybrid algorithm shows different classification optimization results using different GA (crossover and mutation types) and KNN (the number of k-neighbours) parameter set for applied data mining task.

The GA/KNN hybrid algorithm shows very good and stable classification error optimization results using both weights *w* approaches (each weight value w is real parameter from [0; 1] or the sum of all weight values *w* that is real parameter from [0; 1] equals to 1) and the corresponding to these approaches crossover and mutation operators.

References

1. Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning. - New York: Addison Wesley, 1989. - 432 pages.
2. Han J., Kamber M. Data Mining: Concepts and Techniques, 2nd Edition. - Morgan Kaufmann Publishers, 2006. - 800 pages.
3. Lovtsova I. Weight Optimization for Loan Risk Estimation with Genetic Algorithm // Computational Intelligence, Theory and Applications. Proceedings of International Conference 9th Fuzzy Days in

Dortmund, Germany, Sept. 18-20, 2006. Vol. 38. – Berlin Heidelberg: Springer, 2006. - P. 215 - 221.
4. Lovtsova I., Aleksejeva L. Direction Control of Mutation in Genetic Algorithm Used for Function Optimization // International Journal of Academy of Humanities and Economics, Lodz, Poland in cooperation with IEEE Computational Intelligence Society Poland Chapter, 2006. – P. 577 - 584.
5. Lovtsova I., Aleksejeva L. Search Direction Control in Optimization Task Using Genetic Algorithm // International Conference on Operational Research: Simulation and Optimization in Business and Industry. Tallinn, Estonia, May 17-20, 2006. – Kaunas: Technologija, 2006 -P. 114-118.
6. Lovtsova I., Aleksejeva L. Study of Crossover and Mutation in Real Coded Genetic Algorithm Used for Constrained Optimization // Eighth International Conference on Application of Fuzzy Systems and Soft Computing Proceedings, Helsinki, Finland, September 1-3, 2008. – b - Quadrat Verlag, 86916 Kaufering, 2008. – P. 149 – 157.
7. Machine Learning Repository, Statlog (German Credit Data) Data Set URL: http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29. – Visit date September 2009.
8. Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs, 3rd, rev. and extended Edition. - Springer, 1996. - 387 pages.
9. Wright A.H. Genetic Algorithms for Real Parameter Optimization // Foundations of GA. Gregory J.E. Rowlins, ed., Morgan Kaufmann Publishers, 1991. – P. 205-218.

Irina Provorova, Ph.D. student, Riga Technical University, Department of Modelling and Simulation, 1 Kalku Street, Riga, LV - 1658, Latvia
lovcova@inbox.lv.
Irina Provorova received her MSc degree in Information Technology Engineering from Riga Technical University in 2003. Currently she is working on her Ph.D. thesis at the Department of Modelling and Simulation of Riga Technical University. Her research interests include genetic algorithms, genetic operators, their modification and hybridization, as well as their application to solve optimization and data mining tasks.

Serge Parshutin, Ph.D. student, Riga Technical University, Department of Modelling and Simulation, 1 Kalku Street, Riga, LV - 1658, Latvia
serge.parshutin@rtu.lv.
Serge Parshutin received his MSc degree in information technology from Riga Technical University in 2006. Now he is a PhD student at the Faculty of Computer Science and Information Technology and a Lecturer with the Department of Modeling and Simulation at the Riga Technical University. His research interests include data mining and knowledge extraction, intelligent information systems, evolutionary computing and decision support.

Sergejs Provorovs, Mg.sc.ing., Riga Technical University, Department of Modelling and Simulation, 1 Kalku Street, Riga, LV - 1658, Latvia
sprovorovs@inbox.lv.
Sergejs Provorovs received his MSc degree in Electronics and Telecommunications from Riga Technical University in 2006. Now he is the first year MSc student (Information Technology Engineering program) at Riga Technical University. His research interests include evolutionary computing, performing experiments with various parameters and another algorithm combination, and investigation of its working results.

Irīna Provorova, Sergejs Paršutins, Sergejs Provorovs.
Ģenētiskā algoritma pielietojums datu ieguves uzdevumam svaru optimizēšanai
Rakstā izskatīta ģenētiskā algoritma (GA) pielietošana datu ieguves uzdevumam svaru optimizēšanas nolūkos. Datu ieguves uzdevumiem raksturīgs liels datu ierakstu un atribūtu skaits, kuru parasti nepieciešams apstrādāt, veidojot, piemēram, klasifikācijas likumus. Tā rezultātā, lielu datu ierakstu un atribūtu skaita dēļ klasisko metožu klasifikācija notiek ar lielu kļūdu. Šo problēmu risināšanai pielietots ģenētiskais algoritms, kura uzdevums ir atrast tādus svarus katram atribūtam, kas nodrošinātu klasifikācijas kļūdas samazināšanu.

Par klasisko metodi tiek izvēlēts k-tuvāko kaimiņu (KNN) klasifikators un svaru optimizēšanai tiek pielietots modificēts ģenētiskais algoritms. Balstoties uz ģenētiskā un k-tuvākā kaimiņa algoritmu kopējās pielietošanas bāzes, izstrādāts GA/KNN algoritma hibrīds. Rezultātā piedāvāts algoritma hibrīds nodrošina stabilu klasifikācijas kļūdas samazināšanu neatkarīgi no ierakstu un atribūtu skaita, un izvēlētā tuvāko kaimiņu skaita. Ģenētiskā algoritma blokā modificēti krustošanas un mutācijas operatori strādā ar vienādu intensitāti un nodrošina no indivīdu pasliktināšanas.

Ирина Проворова, Сергей Паршутин, Сергей Проворов.
Применение генетического алгоритма для оптимизации весов в задаче извлечения данных
В статье рассматривается применение генетического алгоритма (ГА) для оптимизации весов в задаче извлечения данных. Для задач извлечения данных характерно большое количество записей и атрибутов, которые необходимо обработать, например, при помощи классификационных правил. В результате из-за большого количества записей и атрибутов классический метод дает высокую ошибку при классификации. Для решения данной проблемы был применен генетический алгоритм, задача которого состоит в том, чтобы подобрать такие веса для каждого атрибута, которые обеспечили бы уменьшение ошибки класификации.

В качестве классического метода выбран классификатор к-ближайших соседей (KNN), а для оптимизации весов применен модифицированный генетический алгоритм. На базе совместного применения упомянутых алгоритмов разработан гибридный алгоритм GA/KNN. В результате предложенный гибридный алгоритм обеспечивает стабильное снижение ошибки классификации независимо от количества записей и атрибутов, и выбранного числа ближайших соседей. В блоке генетического алгоритма модифицированные операторы скрещивания и мутации работают с одинаковой интенсивностью и обеспечивают защиту против ухудшения индивидов.