Scientific Journal of Riga Technical University
Computer Science. Information Technology and Management Science
2011
Volume 49

Use of Linear Genetic Programming and Artificial Neural Network Methods to Solve Classification Task

Sergejs Provorovs[1], Arkady Borisov[2], [1-2]*Riga Technical University*

***Abstract* - This paper presents a comparative analysis of linear genetic programming and artificial neural network methods to solve classification tasks. Usually classification tasks have data sets containing a large number of attributes and records, and more than two classes that will be processed using, for example, created classification rules. As a result, by using classical method to classify a large number of records, a high classification error value will be obtained. The artificial neural networks are often used to solve classification task, mostly obtaining good results. The linear genetic programming is a new direction of evolution algorithms that is not widely researched and its application areas are not well defined. However, some advantages of linear genetic programming are based on genetic operators whose structure does not require complicated calculations.**

During this work approximately 400 experiments were conducted with linear genetic programming and artificial neural network methods, using various data sets with different quantity of records, attributes and classes.

Based on the results received, conclusions on possibilities of using the methods of linear genetic programming and artificial neural networks in classification problems were drawn, and suggestions for improving their performance were proposed.

***Keywords* – linear genetic programming, artificial neural networks, classification task, cross-validation**

I. Introduction

This paper addresses linear genetic algorithm (LGP) and artificial neural networks (ANN) application to solve the classification task. Nowadays, classification tasks are very popular and their complicity increases according to data amount, record variable (often they are very noisy) and needed classifying number of class (more than two). The simplest kind of classification problems is to identify some object as a member of a known class. Typically, these classes are stereotypes that are hierarchically organized, and the process of identification is one of matching observations of an unknown entity against features of known classes – for example, plant (Iris), animal (Zoo) or object (Glass) statistical database, using a definition of features, such as structure, size, colour etc. Also there exists a whole area of classification tasks where the reviewed problem has some exceptions; some data may belong to some classes etc. [7]. There are many different approaches to solve classification tasks - one of the most popular is to apply artificial neural networks that are researched and developed as well. Another approach that was used in this paper is linear genetic programming that is not researched and developed that well, but has implementation advances, inherited from evolutionary algorithms [6].

This paper presents a comparative analysis of linear genetic programming and artificial neural network methods to solve classification tasks. The obtained experimental results were evaluated jointly and compared.

II. Artificial Neural Network

Artificial neural networks (ANN) process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing neurons working in parallel to solve a specific problem. Artificial neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully, otherwise useful time is wasted or even worse - the network might be functioning incorrectly. The disadvantage is as follows: because the network finds out how to solve the problem by itself, its operation can be unpredictable [1].

The Hopfield network consists of a set of N interconnected neurons, which update their activation values asynchronously and independently of other neurons. All neurons are both input and output neurons. The activation values are binary. Originally Hopfield chose activation values of 1 and 0 but using values +1 and -1 presents some advantages discussed below [6].

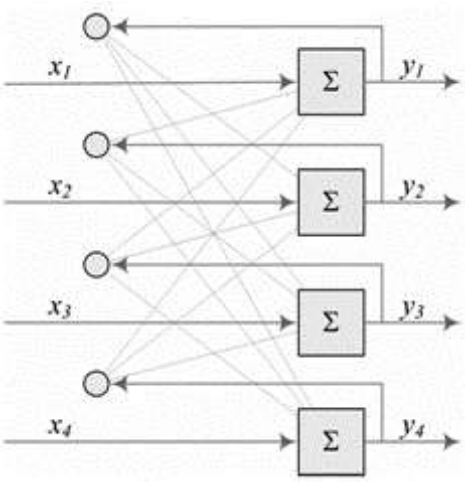

Fig. 1. ANN with back-propagation

III. LINEAR GENETIC PROGRAMMING

In recent years different types of genetic programming have emerged. They all follow the basic idea of GP to automatically evolve computer programs. Three basic forms of representation may be distinguished for genetic programs. Besides the traditional tree representations, these are linear and graph representations [2].

Linear genetic programming (LGP) is a GP variant that evolves sequences of instructions from an imperative programming language or from a machine language.

In Linear Genetic Programming we can identify two spaces. The phenotype space is a set of mathematical functions and the genotype space is a set of programs in a certain representation. These programs are composed from elements of the used programming language. LGP programs are represented as instruction sequences that when executed give us the solution to our problem. For this work, the instructions can have two or three registers. Each instruction is composed of (see Fig. 1):

- A destination register (output) that stores the result;
- One or two registers that store the values used in the operation;
- An operation from the operation set defined.

```
L0: f[0]+=Input2;
L1: f[1]+=f[0];
L2: f[0]=cos(f[0]);
L3: f[0]=cos(f[0]);
L4: f[0]=cos(f[0]);
L5: f[1]*=f[0];
L6: f[1]+=f[0];
L7: f[0]*=Input4;
L8: f[0]*=f[0];
L9: f[0]-=f[1];
L10: f[0]+=-0.9765300750732422f;
L11: f[0]/=Input3;
L12: f[0]=cos(f[0]);
L13: f[0]*=pow(2,TRUNC(f[1]));
L14: f[0]=cos(f[0]);
L15: f[0]*=f[0];
L16: f[0]*=f[0];
L17: f[0]=cos(f[0]);
L18: f[0]*=Input4;
L19: f[0]*=f[0];
L20: f[0]-=f[1];
L21: f[0]*=Input4;
L22: f[0]/=Input3;
L23: f[0]=cos(f[0]);
L24: f[0]+=Input4;
```

Fig. 2. The individual in LGP

The LGP uses the following steps to evolve a computer program that predicts the target output from a data file of inputs and outputs [4]:

- Initializing a population of randomly generated programs;
- Running a selection. In this step four programs are selected from the population randomly. They are compared and based on fitness, two programs are chosen as winners and two as losers;
- Transforming the winner programs. After that two winner programs are copied and transformed probabilistically as follows:
  a) parts of the winner programs are exchanged with each other to create two new programs (crossover operation);
  b) each of the selection winners are exchanged randomly to create two new programs (mutation operation);
- Replacing the loser programs in the selection with the transformed winner programs. The winners of the selection remain without change;
- Repeating steps two to four until convergence. A program defines the output of the algorithm that simulates the behaviour of the problem to an arbitrary degree of accuracy.

The described LGP steps is a very time-consuming process that cannot be implemented manually, so, to provide calculation, it was decided to use the following programs – existing program „Discipulus" and specially developed programs „Data Sets Maker" and „Testing program for LGP and ANN methods".

In linear GP a user-defined number of variable registers, the register set, is made available to a genetic program. Besides the minimal number of input registers required to hold the program inputs before execution, additional registers can be provided in order to facilitate calculations. Normally these so-called calculation registers are initialized with a constant value each time a program is executed on a fitness case. The instruction set defines the particular programming language that is evolved [5]. Two-operand instructions may either possess two indexed variable (registers) operands or one indexed variable and a constant. One-operand instructions use only register operands. If there cannot be more than one constant per instruction, the percentage of instructions holding a constant is equal to the proportion of constants in programs. This is also the selection probability of a constant operand during initialization of programs and during mutations [3].

TABLE I
INSTRUCTION SET

Instruction type	**General notation**
Arithmetic operations	$r_i \coloneqq r_j + r_k$ $r_i \coloneqq r_j - r_k$ $r_i \coloneqq r_j \cdot r_k$ $r_i \coloneqq \frac{r_j}{r_k}$
Exponential functions	$r_i \coloneqq r^{r_k}$ $r_i \coloneqq e^{r_j}$ $r_i \coloneqq \ln(r_j)$ $r_i \coloneqq r_j^2$ $r_i \coloneqq \sqrt{r_j}$
Trigonomic functions	$r_i \coloneqq \sin(r_j)$ $r_i \coloneqq \cos(r_j)$
Boolean operations	$r_i \coloneqq r_j \wedge r_k$ $r_i \coloneqq r_j \vee r_k$ $r_i \coloneqq \neg\ r_k$
Conditional branches	$if\ (r_i > r_k)$, $if\ (r_i \geq r_k)$ $if\ (r_i < r_k)$, $if\ (r_i \leq r_k)$ $if\ (r_i = r_k)$

In order to minimize the input range assigned to a function value, undefined negative inputs have been mapped to defined absolute inputs. This permits evolution to integrate protected instructions into robust program semantics more easily [2].

LGP Evolutionary Operators. LGP evolutionary operators are very similar to evolutionary operators - reproduction, crossover (linear crossover was used during experiment for this paper) and mutation have been used in this research. Reproduction involves copying, unchanged, the selected program to the next generation (selection operator).

The linear crossover operator involves two parents. A random sequence of instructions in the first parent is selected and replaced with a random sequence of instructions from the second parent. In linear GP, for record, crossover points may not be selected inside an instruction and mutations may not exchange an instruction operator for a register [2]. The resulting program is the offspring. If the newly produced program is longer than the maximum length allowed, then an instruction is randomly selected and removed until the program can fit into the maximum length. An example of linear crossover is shown in Fig. 3 [2].

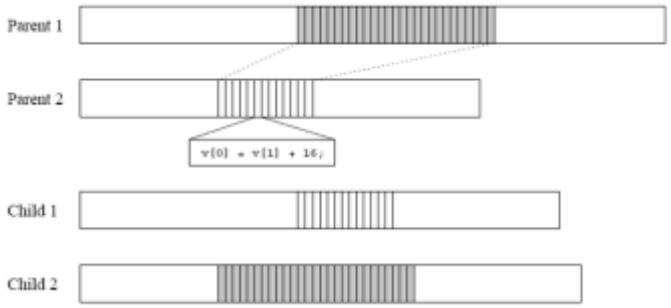

Fig. 3. Crossover operator in LGP

Mutation of the imperative representation resulting in the exchange of a register may have a large effect on the functional program structure and on data flow. Even if the absolute program structure is altered only slightly, the effective program may change dramatically. Many instructions preceding the mutated instruction may be deactivated or reactivated. These minimum mutations are possible due to weaker constraints of the functional structure and due to the existence of non-contiguous graph components in linear programs [4].

LGP fitness function. Fitness of an individual program is computed by an error function on a set of input-output examples. These so-called fitness cases define the problem that should be solved or approximated by a program. A squared error function penalizes larger errors more heavily than smaller errors. Equation 1 defines a related measure, the mean square error (MSE) [2]:

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(i_k - o_k)^2, \quad (1)$$

where MSE – fitness function (mean square error);
i_k – the known (input) value of data set;
o_k – the evaluated (output) value of data set.

Detection and deletion of introns. Non-effective code in genetic programs which is referred to as “intron” represents instructions without any influence on the program behaviour. Structural introns act as protection that reduces the effect of variation on the effective code and also allow variations to remain neutral in terms of fitness change [4]. The imperative program structure in linear GP permits structurally no effective instructions to be identified efficiently. It allows the corresponding effective instructions to be extracted from a program during runtime and to be copied to a temporary program buffer once before the fitness of the program is calculated [2].

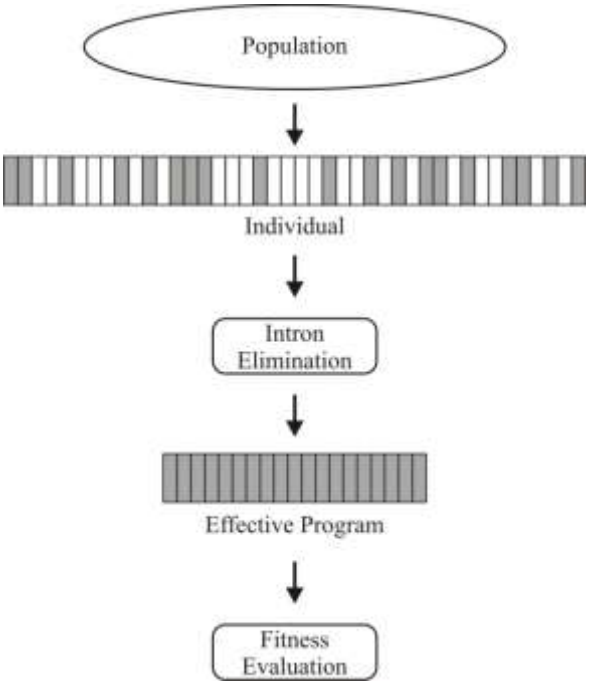

Fig. 4. Detection and deletion of introns

Thereby, the representation of individuals in the population remains unchanged while valuable computation time for non-effective code is saved. No potentially relevant genetic material is lost and intron code may play its role during the evolutionary process. By analogy to the elimination of introns in nature, the linear genetic code is interpreted more efficiently [8].

IV. EXPERIMENTS

The experiments were made using five databases from „UCI Machine Learning Repository" with different number of attributes, classes and records (see Table II) [9].

The data are represented as numerical and categorical types and also as edited with several indicator variables that are added to make them suitable for algorithms.

TABLE II
DATABASES

No.	Database	Number of attributes	Number of classes	Number of records
1	Iris	4	3	150
2	Glass	9	7	214
3	Heart Statlog	13	2	270
4	Zoo	17	7	101
5	Abalone	8	29	4177

To make experiments on the described five statistical data sets, an existing („Discipulus" [10] and Multiple Back-Propagation" [11]) and specially developed programs were used. To learn the classification competence of LGP and ANN methods, it was decided to provide a number of experimental series using data sets with different record and class numbers and combination – the total number of experiments is 400. An application of the following algorithm (see Fig. 5) helps to compare the efficiency of LGP and ANN methods classification result (it was applied during classification to each of the described database).

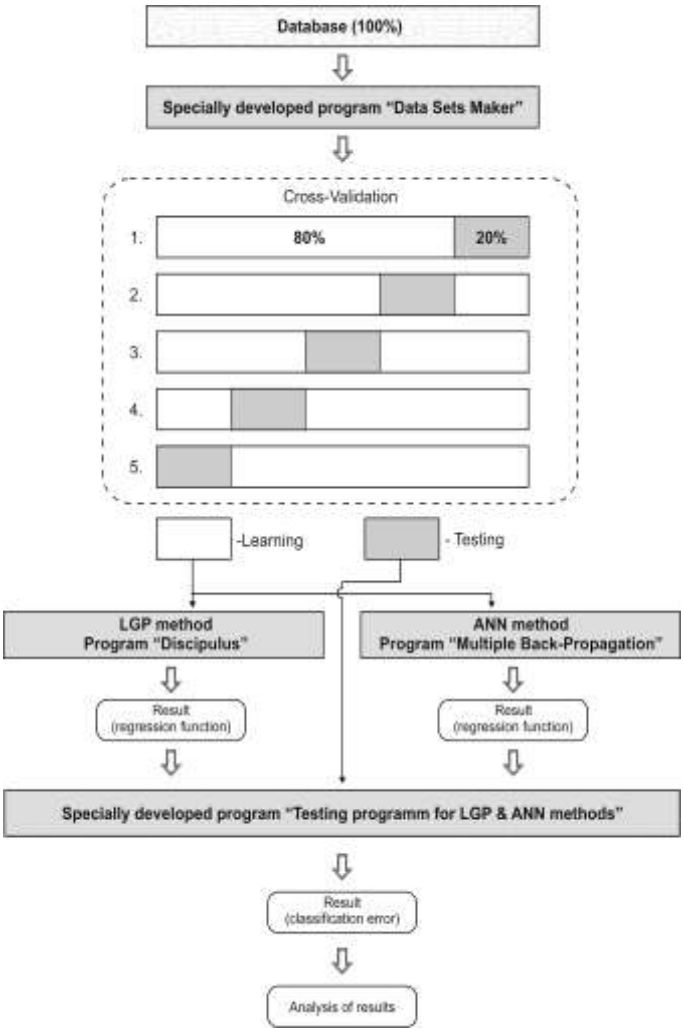

Fig. 5. LGP and ANN results for testing algorithm

The description by steps of the visualised LGP and ANN results testing algorithm (see Fig. 5) is as follows:

1. create a training data set (take 80% of records from the used data set) using specially developed program „Data Sets Maker";
2. create a test data set (take 20% of records from the used data set) using specially developed program „Data Sets Maker";
3. load the training data set in a special existing program (for LGP method the „Discipulus" [10] and for ANN method „Multiple Back–Propagation" [11] programs were used) for performing the training and creating the regression function;
4. adapt the result of the applied program (the created regression function) to a specially developed testing

program „Testing program for LGP and ANN methods";

5. calculate the classification error of the used methods (LGP and ANN) for test data set using a specially developed program „Testing program for LGP and ANN methods";
6. apply the cross-validation approach, going to Step 1;
7. calculate the mean result value for each method.

As an additional stopping criteria, it was decided to use time constraint – maximum time limit for one experiment that means the best decision finding time would not be longer than 10 minutes.

The estimation of LGP algorithm implementation is based on comparison with the ANN classification result, namely - the classification error. To solve the described task, the following experiments were performed (see Table III).

TABLE III
THE LGP PARAMETERS OF EXPERIMENTS

Parameter name	**Parameter value**
Population size	500
Initial program length	80
Maximal program length	512
Crossover rate	90 %
Mutation rate	10 %
Constant rate (from the attribute number)	50 %

To make a comparison of working results of LGP and ANN methods, a whole series of experiments were made. Each experiment group used five data bases that differed by their data classification difficulties.

V. EXPERIMENTAL RESULTS

The results of all experiments produced by the LGP and ANN were evaluated together, grouping by the used data set (see Table IV).

TABLE IV
EXPERIMENTAL RESULTS OF LGP AND ANN

Database	**Classification Error of LGP, %**	**Classification Error of ANN, %**
Iris	3,80	4,72
Glass	27,84	30,50
Heart Statlog	22,90	21,26
Zoo	11,46	13,86
Abalone	60,84	45,70

As the results of the table show, classification errors depend on the class number – higher class number gives higher classification error. Also the attribute number has an influence on the size of classification error. So, to obtain impartial result for LGP and ANN, it was decided to reduce class number by making it the same for each data base.

The results of the experiments with the same class number produced by the LGP and ANN algorithms are evaluated together and summarised in Table V.

TABLE V
EXPERIMENTAL RESULTS OF LGP AND ANN USING TWO CLASSES

Database	**Classification Error of LGP, %**	**Classification Error of ANN, %**
Iris	2,15	3,27
Glass	11,78	15,82
Heart Statlog	22,36	20,11
Zoo	5,03	7,98
Abalone	28,17	27,72

Experimental results were compared using database classification into different classes (Iris - 2; Glass – 7; Heart Statlog – 2; Zoo – 7; Abalone – 29) and into two classes only; the obtained result had much better visible improvement on how large the difference in classes was. If the difference in the class number is slight or there is no difference at all (Heart Statlog database), the results also do not have impressive advancement or changes, respectively.

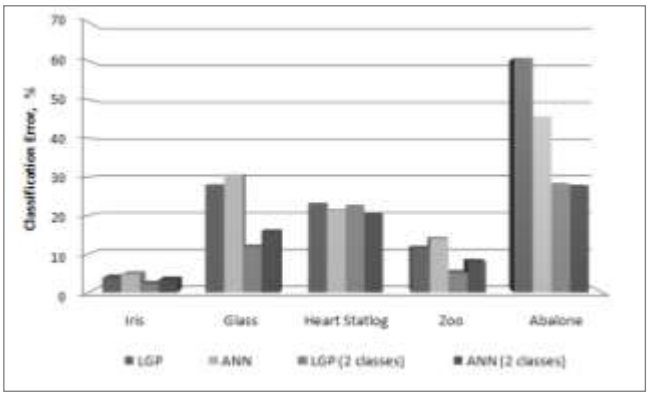

Fig. 6. Experimental results of LGP and ANN (different class number and class number = 2)

In general, the results obtained show that in some experiments LGP is better in solving this kind of classification task and using the described parameters than ANN.

VI. CONCLUSIONS

In most of the experiments performed, the LGP algorithm provides better classification results than the ANN algorithm – the classification error is not visibly reduced, it also depends on the classifying database parameters, like the number of classes, records and attributes. By reducing class number to two classes for each database, the classifying error also

decreases both for LGP and ANN. This insufficient effectiveness of LGP in comparison with ANN could be explained by using the program „Discipulus" that employs standard evolutionary operators and is not meant for an individual tuning for each task (database classifying). Therefore, the LGP parameters and operators need to be tuned – especially the evolution operators that have the type variety.

REFERENCES

[1] Takahaši A. Neural Networks in Fingerprint Classification Problem // Scientific Proceedings of Riga Technical University, Series 5, Vol. 36 (2008), pp. 83.-92.
[2] Brameier M., Banzhaf W. Linear Genetic Programming – USA: Springer, 2007.
[3] Brameier M., Banzhaf W. A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining // Evolutionary Computation, Issue 1, Vol. 5 (2001), pp. 17-26.
[4] Brameier, M. and W. Banzhaf. Linear Genetic Programming. Springer: New York, 2007.
[5] Fogelberg C., Zhang M. Linear Genetic Programming for Multi-class Object Classification // Computer Science, vol. 38, (2005), pp. 369-379.
[6] Hasan Örkcü H., Comparing performances of back-propagation and genetic algorithms in the data classification // Expert Systems with Applications, Issue 4, Vol. 38 (2011), pp. 3703-3709.
[7] Norvig P., Russell S.J. Artificial Intelligence – A Modern Approach. Second Edition – New Jersey: Prentice Hall, 2003.
[8] Tan, P.-N.,Steinbach, M., Kumar, V. Introduction to Data Mining. Boston, MA: Pearson Education Inc., 2006.
[9] Machine Learning Repository, Data Sets, 2007. [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html [Accessed: Sept. 15, 2009].
[10] RML Technologies, Free Software: Discipulus™ 5 Genetic Programming, 2011. [Download]. Available: https://www.rmltech.com/store/store.aspx [Accessed: Aug. 20, 2009].
[11] Multiple Back-Propagation, Free Software: Multiple Back-Propagation, 2011. [Download]. Available: http://dit.ipg.pt/MBP/Download.aspx [Accessed: Aug. 20, 2009].

Sergejs Provorovs is a Doctoral student at the Institute of Information Technology, Riga Technical University. He received his MSc degree in Information Technology from Riga Technical University in 2011. His research interests include evolutionary computing, performing experiments with various parameters and another algorithm combination, and investigation of its working results.

Arkady Borisov has a Doctoral degree in Technical Sciences in the field of Control in Technical Systems and the Dr.habil.sci.comp. degree.
He is Professor of Computer Science at the Faculty of Computer Science and Information Technology, Riga Technical University (Latvia). His research interests include fuzzy sets, fuzzy logic and computational intelligence. He has 210 publications in the field.
He has supervised a number of national research grants and participated in the European research project ECLIPS.

Sergejs Provorovs, Arkadijs Borisovs. Lineārās ģenētiskās programmēšanas un mākslīgā neironu tīkla pielietošana klasifikācijas uzdevuma risināšanai
Rakstā izskatīta lineārās ģenētiskās programmēšanas un mākslīgā neironu tīkla metožu pielietošanas salīdzinošā analīze klasifikācijas uzdevuma risināšanā. Klasifikācija ir viens no aktīvi pētītiem un pielietotajiem virzieniem. Parasti klasifikācijas uzdevuma datu kopa satur lielu atribūtu un ierakstu skaitu, kā arī vairāk par divām klasēm, kurās jāklasificē dati, pielietojot izveidotos klasifikācijas likumus. Rezultātā klasifikācijas metodes pielietošana lielā ierakstu skaita dēļ, dod lielu klasifikācijas kļūdu. Mākslīgais neironu tīkls bieži tiek pielietots klasifikācijas uzdevuma risināšanai, kas vairākos gadījumos nodrošina labus rezultātus. Lineārā ģenētiskā programmēšana ir jauns evolucionāro algoritmu virziens, kas vēl nav labi izpētīts un tā pielietošanas sfēras vēl nav labi definētas. Tomēr var atzīmēt dažas lineārās ģenētiskās programmēšanas priekšrocības, kas balstās uz ģenētiskajiem operatoriem, kuri pēc savas struktūras nesatur sarežģītus aprēķinus. Darba gaitā tika veikti 400 klasifikācijas uzdevuma risināšanas eksperimenti, pielietojot minētās metodes, izmantojot piecas datu kopas ar dažādu atribūtu un klašu skaitu. Eksperimentu realizācijai tika izmantota speciālā programmatūra – divas esošās („Disciplinus" un "Multiple Back-Propagation") un divas speciāli izstrādātās („Testing program for LGP and ANN methods" and „Data Sets Maker"). Lai nodrošinātu objektīvus eksperimentu rezultātus, eksperimentos tika pielietota šķērsvalidācija katrai datu kopai.
Pamatojoties uz iegūtajiem rezultātiem, tika veikti secinājumi par lineārās ģenētiskās programmēšanas un mākslīgo neironu tīklu metožu izmantošanas iespējām klasifikācijas uzdevumos, kā arī izvirzīti priekšlikumi metodes darbības uzlabošanai un pilnveidošanai.

Сергей Проворов, Аркадий Борисов. Применение методов линейного генетического программирования и искусственных нейронных сетей в задачах классификации
В статье проведен сравнительный анализ применения методов линейного генетического программирования и искусственных нейронных сетей в задачах классификации. Задача классификации является одним из активно исследуемых и применяемых направлений. Обычно данные в задачах классификации содержат много записей, атрибутов и классов, на которые необходимо классифицировать данные, применяя созданные правила. В результате применения методов классификации при большом объеме данных появляется большая ошибка классификации. Искусственные нейронные сети часто применяются для решения задач классификации, что в большинстве случаев обеспечивает хороший результат. Линейное генетическое программирование на данный момент является еще новым, плохо изученным направлением. Тем не менее, необходимо отметить некоторые преимущества линейного генетического программирования, базирующиеся на основе генетических операторов, которые по своей структуре не содержат сложных вычислений.
В ходе работы проведено 400 экспериментов с данными методами, используя различные наборы данных с разным количеством записей, атрибутов и классов. Для проведения экспериментов использовано специальное программное обеспечение – два готовых пакета („Disciplinus" и "Multiple Back-Propagation"), а также дополнительно были разработаны две программы („Testing program for LGP and ANN methods" и „Data Sets Maker").
На основании полученных экспериментальных результатов сделаны выводы о возможностях использования методов линейного генетического программирования и искусственных нейронных сетей в задачах классификации, а также выдвинуты предложения по улучшению работы методов.