Induction of Classification Rules with Grammar-Based Genetic Programming

Pedro G. Espejo, Cristóbal Romero, Sebastián Ventura, César Hervás
Department of Computer Science and Numerical Analysis,
University of Cordoba,
14071 Cordoba, Spain
Email: {pgonzalez, cromero, sventura, chervas}@uco.es

***Abstract*— This paper presents an analysis of the suitability of grammar-based genetic programming for the classification task in data mining. The evolutionary technique is compared with several classic algorithms for inducing decision trees and rules, using classification accuracy as the comparison criterion.**

I. Introduction

Data mining (DM) consists of the extraction of useful, comprehensible and previously unknown knowledge, from huge amounts of data stored in different formats [16]. Classification is one of the most studied problems by DM and machine learning (ML) researchers. It consists in predicting the value of a (categorical) attribute (the class) based on the values of other attributes (the predicting attributes). In the ML and DM fields, classification is usually approached as a supervised learning task. A search algorithm is used to induce a classifier from a set of correctly classified data instances, called the train set. Another set of correctly classified data instances, known as the test set is used to measure the quality of the classifier obtained after the learning process. Different paradigms have been used in order to tackle classification: decision trees [10], inductive learning [8], instance-based learning [1] and, more recently, artificial neural networks [18] and evolutionary algorithms [4]. In this paper, we focus on decision tree, rule induction and evolutionary techniques.

Decision tree methods use greedy algorithms. These algorithms are generally fast, very effective, accurate and able to classify data completely. Most decision tree methods use recursive partitioning techniques that split the data space. However, the greedy nature of these algorithms can overlook multivariate relationships that can't be found when attributes are considered separately. Rule induction algorithms usually employ a specific-to-general approach, in which rules are generalized (or specialized) until a satisfactory description of each class is obtained. Finally, evolutionary algorithms (EA) are based on the use of probabilistic search algorithms inspired by certain points of the Darwinian theory of evolution. The flexibility and robustness of EAs allow the discovery of complex relationships that are usually missed by other algorithms.

In addition to the learning algorithm, another important issue that must be considered in classification is the representation formalism. Rules are one of the most often used formalisms used to represent classifiers, and is the one we have chosen for our work (a decision tree can be easily converted into a rule set [12]). The rule antecedent (IF part) contains a combination of conditions on the predicting attributes, and the rule consequent (THEN part) contains the predicted value for the class. This way, a rule assigns a data instance to the class pointed out by the consequent if the values of the predicting attributes satisfy the conditions expressed in the antecedent, and so, a classifier is represented as a rule set. The rules used in our work have the following format.

<classification rule> ::=
    IF <antecedent> THEN <consequent>
<antecedent> ::=
    <condition> |
    <condition> AND <condition>
<consecuent> ::=
    IS A <class label>
<condition> ::=
    <attribute> <rel operator> <value>
<attribute> ::=
    Predicting attribute
<rel operator> ::=
    $= \mid \neq \mid < \mid > \mid \leq \mid \geq$
<value> ::=
    Value from the corresponding domain
<class label> ::=
    Value from the class domain

The evolutionary algorithm analysed in our work is a variant of genetic programming known as grammar-based genetic programming(GGP) [15]. In GGP a grammar is defined, and the evolutionary process proceeds guaranteeing that every individual generated is legal with respect to the grammar. Besides, the grammar can be used to define different characteristics of the classifiers, and can improve efficiency, since the search space is restricted. Our aim is to go more deeply into the use of GGP for DM tasks. Whereas EAs are today extensively applied to DM [4], this trend is mostly oriented to the use of genetic algorithms (GA), and just a minority of this work makes use of GP, not to mention GGP (see however [17]).

The rest of this paper is organized as follows. First, the different algorithms used in our work are briefly presented: classic decision tree and rule induction algorithms in Section II and GGP in Section III. Section IV gives a thorough

description of our GGP-based DM system. A description of the experiments carried out and an analysis of the results can be found in Section V. Finally, Section VI presents the conclusions and suggested research directions.

II. Classic Classification Algorithms

Decision tree and rule induction algorithms have been extensively used, first by ML and later by DM researchers, in order to obtain high-quality classifiers.

A. Decision Trees

Decision tree learning systems are easy to use and understand. A decision tree is a set of conditions organized in a hierarchical structure. An instance is classified by following the path of satisfied conditions from the root of the tree until reaching a leaf, which will correspond with a class label. The branches coming from each internal node correspond to different values that can take the attribute represented by the node. Usually, these branches are exclusive, that is, non-overlapping.

Most existing tree induction systems proceed in a greedy topdown fashion. To build a decision tree, it is necessary to find at each internal node a test for splitting the data into subsets. A basic task in tree building is to rank attributes according to their usefulness in discriminating the classes in the data, and then establish the appropriate conditions for each of these attributes.

Many algorithms have been developed following this basic approach. We have employed ID3 and C4.5 in our work, two of the most well-known decision tree algorithms. ID3 [11] is a simple algorithm to build decision trees that uses information as a criterion for selecting the branching attribute of a node. After the branching attribute is selected, the training cases are divided by the different values of the attribute. C4.5 [12] is the improved successor of ID3, and includes several enhancements, including a better handling of nominal attributes with many possible values, working with continuous attributes and a pruning method for overfitting avoidance.

B. Rule Induction

Algorithms of this paradigm can be considered as heuristic state-space search, which is based on the two key notions of state and operator. A state is a description of a problem situation in a given instant, and an operator is a procedure which transforms a state into another. Solving a problem consists in finding a sequence of operators which transforms an initial state into a goal state. In rule induction, a state corresponds to a candidate rule and operators correspond to generalization and specialization operations that transform a candidate rule into another. The choice of the operator to be applied is determined by a heuristic function that evaluates the effectiveness of each operator with respect to the given candidate rule. The obtained rules can cover overlapping data regions, that is, an instance can satisfy the antecedents of several rules.

Many algorithms have been proposed for the induction of classification rules. Three of these algorithms have been chosen to be used in our work: One-R, AQ and CN2. One-R [5] is a simple algorithm that constructs rules trying every possible attribute/value combination, in order to choose the conditions attaining a greater classification accuracy. In AQ [7], a selector relates a variable to a value or set of values, a conjunction of selectors forms a complex, and a disjunction of complexes makes up a cover, which is used as the antecedent of a rule. CN2 [3] was designed as an improvement of AQ, adding a proper handling of noisy instances and eliminating its dependence on specific training instances during the search.

III. Grammar-Based Genetic Programming

The evolutionary algorithms (EA) paradigm is based on the use of probabilistic search algorithms inspired by certain points of the Darwinian theory of evolution [13]. Several different techniques are grouped under the generic denomination of EA. There is a general agreement that the four main branches are: evolution strategies, evolutionary programming, genetic algorithms and genetic programming. The essential features shared by all EAs are:

- The use of a population (a group) of individuals (candidate or partial solutions) instead of just one of them.
- A generational inheritance method. Genetic operators are applied to the individuals of a population to give birth to a new population of individuals (the next generation). The main genetic operators are crossover (recombination) and mutation. Crossover swaps a part of the genetic material of several individuals (usually two of them), whereas mutation randomly changes a little portion of the genetic material of one individual.
- A fitness-biased selection method. A fitness function is used in order to measure the quality of an individual. The better the fitness of an individual, the higher its probability of being selected to take part in the breeding of the next generation of individuals, and so, higher is the probability that its genetic material will survive throughout the evolutionary process.

Genetic programming (GP) is essentially considered as a variant of genetic algorithms (GA) that uses a complex representation language to codify individuals. The most often used representation schema is based on trees, although other options exist [2]. The original goal of GP, as its name implies, was the evolution of computer programs. However, GP is nowadays used to evolve other abstractions of knowledge, like mathematical expressions or rule-based systems, for example. The main difference between GA and GP lies in the fact that GP individuals represent programs,[1] therefore individuals consist not only in data structures, but also in operations. This way, tree individuals are usually seen as parse trees, where leafs correspond to terminal symbols (variables and constants) and internal nodes correspond to non-terminals (operators and

[1]Though the conception of what a program is can vary vastly, as we have just seen.

functions). The set of all the allowed non-terminal symbols is called the function set, whereas the allowed terminal symbols constitute the terminal set. Two conditions must be satisfied to ensure GP can be successfully applied to a specific problem: sufficiency and closure. Sufficiency states that the terminals and non-terminals (in combination) must be capable of representing a solution to the problem. Closure requires that each function of the non-terminal set should be able to handle all values it might receive as input. In practice, we often need to evolve programs that handle values of different types, and this makes difficult to meet the closure requirement.

One of the solutions recently proposed to circumvent the closure problem is based on the utilization of grammars to specify a language to which individuals must adhere. This idea leads to grammar-based genetic programming (GGP). GGP makes use of genetic operators that take into account the grammar, in such a way that it is always guaranteed that every individual generated is legal with respect to the grammar. As well as providing a solution to the closure problem, the use of a grammar offers some other advantages. The grammar can be used to bias the inductive process to give certain characteristics to individuals. In addition, efficiency can be improved, since the grammar restricts the search space.

Two main (similar) approaches to GGP exist, described in [15] and [17] respectively. Both approaches conform to the basic description given above. The main difference between these GGP frameworks lies in the kind of grammar used. Whigham's GGP uses a context-free grammar, while Wong's GGP is in fact a hybrid of GP and inductive logic programming [6] which makes use of a particular kind of grammars known as logic grammars. Logic grammars allow to represent context-sensitive information. In our work, we follow the setting proposed by Whigham in [15].

IV. System Description

We have developed a system that employs GGP to evolve rule-based classifiers. However, a rule set can be interpreted in several ways. When an instance is presented to the classification system, different approaches can be applied to find a matching between the instance and the rule set. We are interested in analysing this point, and therefore we compare two different ways of interpreting a rule set:

- Rules are evaluated sequentially and the process stops with the first rule whose antecedent is satisfied. The instance is assigned to the class pointed out by the consequent of the first matched rule. In this approach, the order of evaluation of rules matters. We order the rules in descending order of the class distribution rate, that is, rules corresponding to majority classes are checked out before the minority ones, so that classification accuracy is maximized.
- Every rule is evaluated, counting the number of rules satisfied per class, and the class with a higher number of satisfied rules is chosen. If a tie occurs between several classes, the one with a greater number of instances (the majority class from among this subset of classes with maximum satisfied antecedent count) is chosen.

We use the grammar to specify which relational operators are allowed to appear in the antecedents depending on the data types of the attributes and to determine the number of rules allowed for each class (see below).

A. Individual Representation

In our GGP system each individual represents a complete classifier, that is, a rule set.[2] While every class is guaranteed to have one rule at least, no other limit exists to the number of rules per class. The format of rules is the one described in Section I. The allowed relational operators are determined by the data type of the attribute. We employ '$>$' and '$\leq$' for numerical attributes and '$=$' and '$\neq$' for categorical attributes.

B. Evolutionary Procedure

We employ a tournament-based selection mechanism that proceeds as follows. First a group of individuals is selected from the population at random. Then for each of the selected individuals, the fitness on the training examples is calculated. The tournament members are sorted on fitness and the best ones get a chance to reproduce and replace the worst ones. The offspring are mutated. The dynamic of the evolutionary process is slightly different from the usual, because the tournament size we employ (50 individuals, in our experiments) is bigger than the customary 2, 4 or so. Age of individuals is taken into account. The age is the number of tournaments in which an individual has taken part, and a maximum age is defined to be one of the system parameters. The individuals in the tournament whose age is greater than or equal to the maximum are removed and labelled "old", and those that remain are called "young". The young individuals are sorted by fitness, so that the best ones are at the top of the list. The old individuals are added back to the bottom of the list. The ordered set of individuals is used to construct a set of "families" each containing four individuals: two parents and two unfit individuals which will be replaced when the parents reproduce. In each family, the parents are crossed over, the resulting offspring overwrite the two unfit individuals and then they are mutated. However, crossover is not performed if the fitness values of the two parents are identical. If they are identical then the second parent is mutated, and the two individuals that would have been replaced by offspring survive.

As usual, crossover swaps subtrees of two parents.[3] The number of swaps performed per crossover event is not fixed, but depends on the number of nodes of the parents. In our system, the crossover probability is the probability of crossover per node, rather than indicate if two individuals will perform crossover or not. Similarly, several mutations occur per mutation event, in a number proportional to the number of nodes of the individual. Several mutation operators are available. The

[2] So we follow the Pittsburgh approach [4].

[3] The subtrees to swap are selected randomly, but always guaranteeing that the offspring will adhere to the grammar.

mutation operator to be applied is randomly selected according to given probabilities. The mutation operators are:

- Point mutation. A node (internal or leaf) is randomly selected to be replaced with a function (of the same arity) or terminal selected at random.
- Macromutation. One of the following operations is randomly selected and applied:
  - Replace subtree: subtree replaced with random subtree.
  - Copy subtree: two independent nodes (subtrees not containing each other) are selected and one subtree is copied, replacing the other.
  - Swap subtrees: as above, but subtrees are swapped.
  - Insert internal: a node is expanded by replacing it with a non-terminal node. One of the branches of the new non-terminal is linked back to the original node, and any remaining branches are expanded with new random subtrees.
  - Delete internal: a node is collapsed as follows. Two nodes are chosen, the second belongs to the subtree of the first and is of the same type. The nodes are reconnected, and any intervening nodes are removed from the tree.

The evolutionary process ends when a predetermined number of tournaments has been performed, and the best individual found in any generation is returned as output.

C. Fitness Function

The fitness function used in this work is classification accuracy, that is, the fraction of correctly classified examples:

$$Acc = \frac{\#Correctly\ classified\ examples}{\#Examples} \qquad (1)$$

D. Grammar

Our system uses context-free grammars. This kind of grammar can be represented as a four-tuple (N, Σ, P, S), where N is the alphabet of nonterminal symbols, Σ is the alphabet of terminal symbols, P is the set of productions and S is the start symbol. The productions are of the form $x \rightarrow y$, where $x \in N$ and $y \in \{\Sigma \cup N\}^*$. Productions of the form

$x \rightarrow y$
$x \rightarrow z$

may be expressed using the disjunctive symbol '|', as

$x \rightarrow y \mid z$

An example grammar is shown below. This grammar has been designed to be used with one of the datasets to which our system has been applied (see Section V-A), the well-known Iris dataset, in which three classes are distinguished: iris-setosa, iris-versicolor and iris-virginica. Besides the class, this dataset has four attributes, all of them numerical (x_1, x_2, x_3, x_4).

$\Sigma = \{$ x_1, x_2, x_3, x_4,
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
true, false$\}$

S → RULEA
RULEB
RULEC

RULEA → if COND then 'Iris-setosa' |
RULEA
RULEA

RULEB → if COND then 'Iris-versicolor' |
RULEB
RULEB

RULEC → if COND then 'Iris-virginica' |
RULEC
RULEC

COND → VAR $>$ VAL |
VAR $\leq$ VAL |
COND and COND |
true | false

VAR → $x_1|x_2|x_3|x_4$

NUM → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

As can be noticed by looking at the shown grammars, the rules produced by the evolutionary process can use any number of attributes in their antecedents. This means that our system is able to perform an implicit feature selection [4], that is, it can select from the available attributes those that are relevant to build a good solution.

V. Computational Experiments

We have compared the performance of ID3, C4.5, OneR, AQ, CN2 and GGP on 7 public domain datasets. ID3 has not been applied to all datasets because it can't handle continuous attributes. For GGP we have used the two rule set interpretation approaches described in Section IV. This leads to two different GGP variants: GGP-1r (one rule, the first whose antecedent is satisfied) and GGP-sr (several rules, counting the number of satisfied antecedents per class). The basic approach we follow consists in estimating the accuracy reached by each algorithm for each dataset. The obtained estimates will allow us to compare the performance of GGP and some widely used classic algorithms, in order to assess the suitability of GGP for classification.

A. Datasets

Experiments have been carried out on 7 public domain datasets. All datasets are taken from the UCI Repository of Machine Learning Databases [9]. The selected datasets present a good variety with respect to different characteristics such as: number of instances, number of classes, number of attributes and data type of the attributes. A description of the used datasets is given in Table I.

TABLE I
DATASETS

Name	Instances	Classes	Attributes	Numerical atts.	Categorical atts.
Breast-wisconsin	683	2	9	9	0
Iris	150	3	4	4	0
Lymphography	148	4	18	0	18
Mushroom	8124	2	22	0	22
Pima	768	2	8	8	0
Satimage	6435	6	39	39	0
Tic-tac-toe	958	2	9	0	9

TABLE II
GGP PARAMETERS

Parameter	Value
Population size	500
No. tournaments	1000
Tournament size	50
Maximum age	6
Stopping criterion	No. tournaments
Crossover prob. (per node)	1/75
Mutation prob. (per node)	1/100

B. GGP parameters

Generally, the terminal set is made up of: the names of the attributes in the dataset, an arbitrary selection of values taken from the domains of the attributes and the logical values true and false. The function set consists of the relational operators (see Section IV-A) and the logical operator AND. The values for the other parameters that control the GGP system are given in Table II.

C. Results

Each algorithm is evaluated using stratified 10-fold cross-validation. The dataset is randomly divided into 10 disjoint subsets of equal size in a stratified way (maintaining the original class distribution), and then, 10 runs are performed. In each run, one of the 10 subsets is used as the test set and the other 9 subsets are combined to form the train set. Each of the 10 iterations of the cross-validation procedure involved a single run of each algorithm. The performance of each algorithm is estimated by averaging classification accuracy (measured from the test set) over the 10 folds. Average accuracies are shown in Table III.

We have performed a statistical analysis in order to assess which algorithms are better than other ones. Analysis of variance allows us to determine whether the mean accuracies are significantly different from each other, and multiple comparison procedures (Bonferroni and Tamhane's T2) tell us which means differ [14]. All statistical tests have been performed at a 0,05 significance level. In Table III a '$+$' result is statistically better than a '$-$' result, and a '$*$' result is statistically better than all the other results. The accuracy corresponding to the best performer is displayed in italics.

For Breast-wisconsin, GGP-sr is the best performing algorithm, and it is statistically better than One-R, but no statistical difference exists with respect to the other algorithms. For the Iris dataset, C4.5, GGP-1r and GGP-sr achieve the same best accuracy, being better than AQ and CN2. For Lymphography, GGP-1r and GGP-sr have different average accuracies, but no statistical difference exists between them. Both algorithms are statistically superior to all the rest. For Mushroom, ID3, C4.5 and AQ present the same accuracy, and they are statistically better than CN2 and GGP-1r. For Pima, C4.5 gives highest performance and it is better than AQ and CN2. For Satimage, C4.5 is the best performer, significantly better than all the rest. Finally, for Tic-tac-toe, AQ offers the best result, statistically better than C4.5, One-R, CN2, GGP-1r and GGP-sr.

To sum up, we can say that in three cases (Breast-wisconsin, Iris and Lymphography) GGP is the best performing algorithm. In other occasion (Pima), GGP is not worst than the best performer. For the Mushroom dataset, GGP-1r is worst than the best performer, but GGP-sr is not. Only for Satimage and Tic-tac-toe both variants of GGP are worst than the best algorithm.

Another point of interest for us is the performance comparison between the two proposed approaches for evaluating a rule set (GGP-1r and GGP-sr). We can see that GGP-sr always gives superior or equal results than GGP-1r, however, results don't differ significantly.

VI. CONCLUSIONS AND FUTURE WORK

The main conclusion we can reach from our experiments is that GGP is a competitive technique for classification, since, generally, it can give superior or equal accuracy when compared with other widely used algorithms.

With regard to the study of different ways of evaluating a rule set (1r and sr), we can conclude that the advantage of evaluating all the rules is minimal when compared with the approach consisting in stopping when the first satisfied antecedent is found. Although the difference between both approaches is not statistically significant, it is clear that GGP-sr always gives higher (or equal) accuracy than GGP-1r, and this difference can be important from a practical point of view, so the sr approach seems preferable when evaluating a rule set.

Besides, we feel that the main advantage of using GGP for classification tasks comes from the use of a grammar. We have used grammars to determine which relational operators can appear in the antecedent of rules, but grammars can be used to easily determine different characteristics of the classifiers to be constructed. In addition, efficiency can be improved, since the grammar restricts the search space. We think grammars

TABLE III
AVERAGE ACCURACY (%)

Dataset	ID3	C4.5	One-R	AQ	CN2	GGP-1r	GGP-sr
Breast-wisconsin	93,99	94,56	91,56$^-$	93,57	92,57	94,99	*96,56*$^+$
Iris		*94,00*$^+$	92,00	81,99$^-$	57,33$^-$	*94,00*$^+$	*94,00*$^+$
Lymphography		36,67	39,33	44,00	3,33	*76,33**	*76,95**
Mushroom	*100,00*$^+$	*100,00*$^+$	98,52	*100,00*$^+$	95,78$^-$	94,74$^-$	98,08
Pima		*74,48*$^+$	72,27	66,67$^-$	65,61$^-$	67,96	68,48
Satimage		*85,77**	59,71	81,34	48,20	66,00	72,79
Tic-tac-toe	94,57	84,54$^-$	69,92$^-$	*95,62*$^+$	86,54$^-$	74,21$^-$	75,66$^-$

offer a great potential for DM tasks that must be studied more deeply.

The work presented in this paper is only a first step in the study of the applicability of GGP for DM. Our aim is to extend the scope of the present work, increasing the number of datasets and taking into account other quality factors for performance comparison, like the size of classifiers and execution time of algorithms. Another interesting avenue to go more deeply in this research way would be to extend the application of GGP to other data mining tasks, like association rule mining or clustering.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support provided by the Spanish Department of Research of the Ministry of Science and Technology under TIC2002-04036-C05-02 (Department of Computer Science, University of Cordoba) Projects. FEDER also provided additional funding. We also thank all the donors and maintainers of the datasets used in this work.

REFERENCES

[1] D. W. Aha, D. F. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[2] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann / dpunkt.verlag, 1998.

[3] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, no. 4, pp. 261–283, 1989.

[4] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, 2002.

[5] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine Learning*, vol. 11, no. 1, pp. 63–91, 1993.

[6] N. Lavrac and S. Dzeroski, *Inductive Logic Programming: Techniques and Applications*. Ellis Horword, 1994.

[7] R. S. Michalski, "On the quasi-minimal solution of the general covering problem," in *Proceedings of the Fifth International Symposium on Information Processing*, 1969, pp. 125–128.

[8] R. S. Michalski, "A theory and methodology of inductive learning," *Artificial Intelligence*, vol. 20, no. 2, pp. 111–161, February 1983.

[9] P. M. Murphy and D. W. Aha, *UCI Repository of Machine Learning Databases*, web, Department of Information and Computer Science, University of California at Irvine, 1994, http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[10] S. K. Murthy, "Automatic construction of decision trees from data: a multi-disciplinary survey," *Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 345–389, December 1998.

[11] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[12] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[13] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, and H. de Garis, "An overview of evolutionary computation," in *European Conference on Machine Learning*, ser. Lecture Notes in Computer Science, P. Brazdil, Ed., vol. 667. Springer, 1993, pp. 442–459.

[14] A. C. Tamhane and D. D. Dunlop, *Statistics and Data Analysis: from Elementary to Intermediate*. Prentice Hall, 1999.

[15] P. A. Whigham, "Gramatical bias for evolutionary learning," Ph.D. dissertation, School of Computer Science - University College - University of New South Wales - Australian Defence Force Academy, 1996.

[16] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

[17] M. L. Wong and K. S. Leung, *Data Mining using Grammar-Based Genetic Programming and Applications*. Kluwer Academic Publishers, 2000.

[18] J. M. Zurada, *Introduction to Artificial Neural Systems*. West Publishing Company, October 1992.