Solving Multiclass Classification Problems by Genetic Programming

Stephan WINKLER
Institute for Formal Models and Verification, Johannes Kepler University
Linz, 4040, Austria

and

Michael AFFENZELLER
Institute for Formal Models and Verification, Johannes Kepler University
Linz, 4040, Austria

and

Stefan WAGNER
Institute for Formal Models and Verification, Johannes Kepler University
Linz, 4040, Austria

ABSTRACT

In this paper a multiclass classification problem solving technique based on genetic programming is presented. Classification algorithms are designed to learn a function which maps a vector of object features into one of several classes; this is done by analyzing a set of input-output examples of the function (also called "training samples"). Here we present a method based on the theory of genetic algorithms and genetic programming that interprets classification problems as optimization problems. The major aspects presented in this paper are suitable genetic operators for this problem class (mainly the creation of new hypotheses by merging already existing ones and their detailed evaluation) we have designed and implemented. We define a novel function for measuring a classificator model's quality that takes into account several different features of the model to be evaluated; an extended version of ROC curves that can be applied not only to two-class-classification but also to multiclass classification problems, is also presented. The experimental part of the paper documents the ability of this method to yield very satisfying results; selected results achieved for two classification benchmark problems are discussed.

Keywords: Genetic Programming, Genetic Programming, Data Mining, Classification, Knowledge Representation

1. A SHORT INTRODUCTION TO EVOLUTIONARY COMPUTATION

Evolutionary computing is the collective name for heuristic problem-solving techniques based on the principles of biological evolution, which are natural selection and genetic inheritance. One of the greatest advantages of these techniques is that they can be applied to a variety of problems, ranging from leading-edge scientific research to practical applications in industry and commerce; by now, evolutionary algorithms are in use in various disciplines like optimization, artificial intelligence, machine learning, simulation of economic processes, computer games or even sociology. The forms of evolutionary computation relevant for the work described in this paper are genetic algorithms (GA) and genetic programming (GP).

Genetic Algorithms
The fundamental principles of GAs were first presented by Holland [7], overviews about GAs and their implementation in various fields were given for instance by Goldberg [4] and Michalewicz [10]. A GA works with a set of candidate solutions (also known as individuals) called population. During the execution of the algorithm each individual has to be evaluated, which means that a value indicating the model quality is returned by a fitness function. New individuals are created on the one hand by combining the genetic make-up of two solution candidates (this procedure is called "crossover" or "recombination"), producing a new "child" out of two "parents", and on the other hand by mutating some individuals, which means that randomly chosen parts of genetic information are changed (normally a minor ratio of the algorithm's population is mutated in each generation).

Beside crossover and mutation, the third decisive aspect of genetic algorithms is selection. In analogy to biology this is a mechanism also called "survival of the fittest". As already mentioned, each individual is associated with a fitness value. The individual's probability to propagate its genetic information to the next generation is proportional to its fitness; the better a solution candidate's fitness value, the higher the probability, that its genetic information will be included in the next generation's population. This procedure of crossover, mutation and selection is repeated many times (over many generations) until some termination criterion is fulfilled.

Genetic Programming

Genetic programming was first explored in depth in 1992 by John R. Koza, a computer scientist at Stanford University, CA, USA. In his famous book "Genetic Programming: On the Programming of Computers by Means of Natural Selection" [8] he pointed out that virtually all problems in artificial intelligence, machine learning, adaptive systems, and automated learning can be recast as a search for a computer program, and that genetic programming provides a way to successfully conduct the search for a computer program in the space of computer programs.

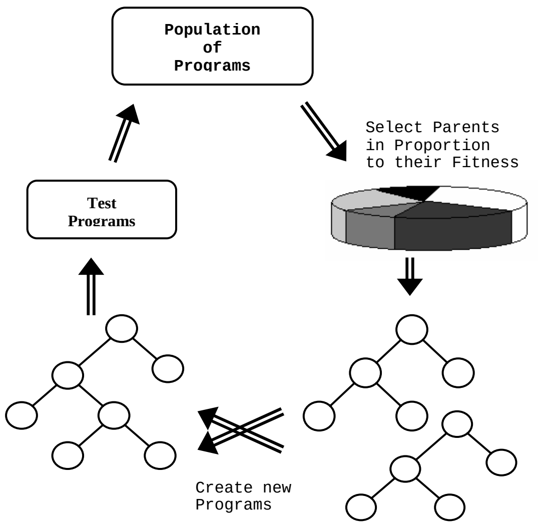

Fig. 1: The genetic programming cycle (taken from [9]).

Similar to GAs, GP works by imitating aspects of natural evolution to generate a solution that maximizes (or minimizes) some fitness function; a population of solution candidates evolves through many generations towards a solution using certain evolutionary operators and a "survival-of-the-fittest" selection scheme. The main difference between the execution of a GA and GP is that, whereas GAs are intended to find an array of characters or integers representing the solution of a given problem, the goal of a GP process is to produce a computer program - or, as in our case, a formula - solving the optimization problem at hand. Typically the population of a GP algorithm contains a few hundred individuals and evolves through the action of operators known as crossover, mutation and selection. The right part of Fig. 1 visualizes the GP cycle: As in every evolutionary process, new individuals (in GP's case, new programs) are created. They are tested, and the fitter ones in the population succeed in creating children of their own. Unfit ones die and are removed from the population [9].

2. DATA MINING AND CLASSIFICATION

In general, data mining is understood as the practice of automatically searching large stores of data for patterns. Nowadays incredibly large (and quickly growing) amounts of data are collected in commercial, administrative, and scientific databases. Several sciences (e.g., molecular biology, genetics, astrophysics and many others) produce extreme amounts of information which are often collected automatically. This is why it is impossible to analyze and exploit all this data manually; what is needed are intelligent computer systems that can extract useful information (such as general rules or interesting patterns) from large amounts of observations. In short, "data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [3].

Classification is understood as the act of placing an object into a set of categories, based on the object's properties. Objects are classified according to an (in most cases hierarchical) classification scheme also called taxonomy. Amongst many other possible applications, examples of taxonomic classification are biological classification (the act of categorizing and grouping living species of organisms), medical classification and security classification (where it is often necessary to classify objects or persons for deciding whether a problem might arise from the present situation or not). A statistical classification algorithm is supposed to take feature representations of objects and map them to a special, predefined classification label. Such classification algorithms are designed to learn (i.e. to approximate the behavior of) a function which maps a vector of object features into one of several classes; this is done by analyzing a set of input-output examples ("training samples") of the function. Since statistical classification algorithms are supposed to "learn" such functions, we are dealing with a specific sub area of Machine Learning and, more generally, Artificial Intelligence.

There are several approaches which are nowadays used for solving data mining and, more specifically, classification problems. The most common ones are (as for example described in [11]) decision tree learning, instance-based learning, inductive logic programming (such as Prolog, e.g.) and reinforcement learning.

Unlike these methods, the approach we have designed is a genetic programming model including appropriate crossover and mutation operators for this problem; this GP based approach is described in the next chapter.

3. A GP-BASED STRUCTURE IDENTIFICATION APPROACH

Preliminary work for the approach presented in this paper was done for the project "Specification, Design and Implementation of a Genetic Programming Approach for Identifying Nonlinear Models of Mechatronic Systems" which is a part of a bigger strategical project at the Johannes Kepler University Linz, Austria. The goal of this project was to find models for mechatronic systems. It was successfully shown ([13], [14], [15]) that methods of GP are suitable for determing an appropriate mathematical representation of a physical system.

We have used the methods implemented for this project for developing a GP-based approach for a statistical classification algorithm. This algorithm works on a set of training examples with known properties [$X_1 \ldots X_n$]. One of these properties (X_t) has to represent the membership information with respect to the underlying taxonomy. On the basis of the training examples, the algorithm tries to evolve (or, as one could also say, to "learn") a solution, i.e. a formula, that represents the function which maps a vector of object features into one of the given classes. In other words, each presented instance of the classification problem is interpreted as an instance of an optimization problem; a solution is found by a heuristic optimization algorithm.

The goal of the implemented GP classification process is to produce an algebraic expression from a database containing the measured results of the experiments to be analyzed. Thus, in the GP approach we have designed and implemented for this project, the GP algorithm works with solution candidates that are tree structure representations of symbolic expressions. These tree representations consist of nodes and are of variable length.

The nodes can either be nonterminal or terminal ones. A nonterminal node signifies a function performing some action on one or more property values within the structure to produce the values of the target property (which of course should be the property which indicates which class the objects belong to). A terminal node represents an input variable (i.e., a pointer to one of the objects' properties) or a constant. The nonterminal nodes have to be selected from a library of possible functions, a pool of potential nonlinear model structural components; the selection of the library functions is an important part of any GP modelling process [5] because this library should be able to represent a wide range of systems. The trees are built by combining nodes according to grammar rules defining the number of inputs for each node type. When the evolutionary algorithm is executed, each individual of the population represents one structure tree; usually the structures are limited by a predefined maximum tree size.

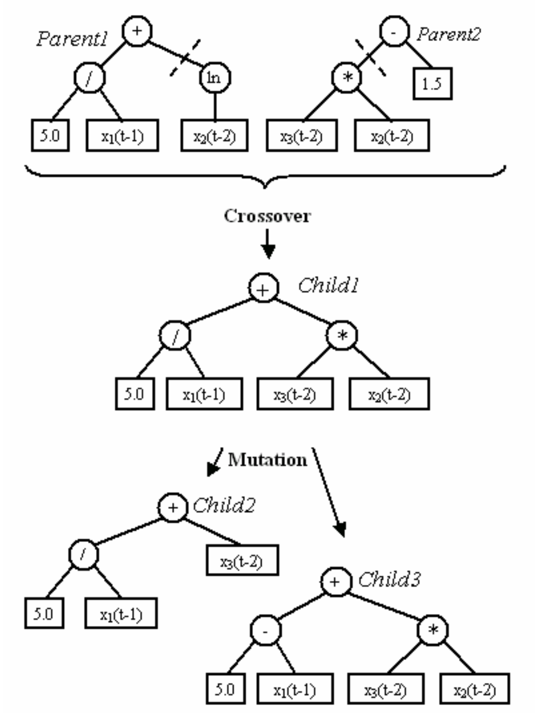

Fig. 2: Examples of genetic operations on tree structures representing formulae: The crossover of parent1 and parent2 yields child1; child2 and child3 are possible mutants of child1.

Since the tree structures have to be usable by the evolutionary algorithm, mutation and crossover operators for the tree structures have to be designed. Both crossover and mutation processes are applied to randomly chosen branches (in this context a branch is the part of a structure lying below a given point in the tree). Crossing over, i.e. merging the genetic information of two trees means randomly choosing a branch in each parent tree and replacing the branch of the tree, that will serve as the root of the new child (randomly chosen, too), by the branch of the other tree. As mutating in the context of genetic algorithms means modifying a solution candidate randomly and so creating a new individual, mutation in the case of identifying structures works by choosing a node and changing it: A function symbol could for instance become another function symbol or be deleted, the value of a constant node or the time offset of a variable could be modified. This procedure is less likely to improve a specific structure but it can help the optimization algorithm to reintroduce genetic diversity in order to restimulate genetic search.

This approach has also been implemented as a part of the already existing “HeuristicLab”, a framework for prototyping and analyzing optimization techniques [12] for which both generic concepts of evolutionary algorithms and many functions to evaluate and analyze them are available. The programming language chosen for this project (and the HeuristicLab) is C# using the Microsoft .NET Framework 1.1. We have also tested our approach intensively. Moreover, in addition to standard GP implementations, new generic concepts [1], based on evolutionary algorithms and developed to increase the quality of the produced solutions, were used and compared to the classical GP approach. Examples of the results of our test series are given in Chapter 5.

4. QUALITY MEASUREMENT OF A CLASSIFICATOR MODEL

Within the approach we present here one also has to deal with the problem that a mathematical formula, which is the output of the classification algorithm, cannot assign a certain class to a given test object. What is done instead is that a value is calculated and, on the basis of this number, one can determine how an object is classified. Furthermore one then has to determine optimal thresholds between the original class values so that as many objects as possible are classified into the correct class (cf. statistical approaches such as logistic regression): A classificator is formed by a classification function in combination with a threshold value.

Since the GP algorithm maximizes or minimizes some objective fitness function (better model structures evolve as the GP algorithm minimizes the fitness function), every solution candidate has to be evaluated. In the context of classification this function should be an appropriate measure of the level of agreement between the calculated class values of the measured objects or experiments and their original ones (i.e., the underlying classification hypothesis and the real world). In detail, the following aspects have to be considered when evaluating a solution candidate:

- Calculating the sum of squared errors J between original class values o_i and calculated class values c_i is a good measurement of the quality of the formula at hand:

$$J = \sum_{i=0}^{N} (o_i - c_i)^2 \qquad (1)$$

- A good model for the classification schema which is to be identified should classify the test sample objects so that the classes can be separated from each other as well as possible. In other words, each test object's calculated class value should be nearer to the object's original class than to any other class value. This can be measured for a class c as follows:

  Let A be the set of objects that originally belong to the given class c and B the set of all other objects. All members of A are compared to all members of B which means that all pairs in $A \times B = C$ have to be considered. If an element b of B has a greater distance to c than an object a in A or either of a and b is a bigger value than c and the other one is not, then this has to be considered as a correct pair. We collect all these correct pairs in *D*. If classes are perfectly separated from each other, then *D=C*. Thus, the separability S_i of a class c_i is calculated as

$$S_i = \frac{|D_i|}{|C_i|} \qquad (2)$$

In fact, this value also corresponds to the area under the so-called Receiver Operating Characteristic (ROC) curve (as described in further detail in the next Section).

- The third interesting property of a classification model is how clearly the classes can be separated from each other. This is necessary since especially in the context of GP based classification studies we have often observed that classificators are produced which separate the given classes quite accurately but not very clearly (the interspace between the estimated class ranges is often not very big). Thus, the quality of the prognosis decreases strongly as we have seen in several test runs.

  ROC curves and the area under them do not consider, how clearly classes can be separated but only how many false classifications are done in the optimal case. This is why we have developed the following approach how to measure this: The ranges R_i of the estimated class values for all members of a given class c_i are calculated. Of course, the smaller the ranges of the calculated class values of the classes' objects are, the more precise the model seems to be and the better the corresponding GP solution candidate has to be evaluated.

At the moment we are trying to find out, how *J*, *S* and *R* have to be weighted (i.e. multiplied with their respective weighting factors *j*, *s* and *r*) in order to maximize the quality of the results produced by the GP algorithm.

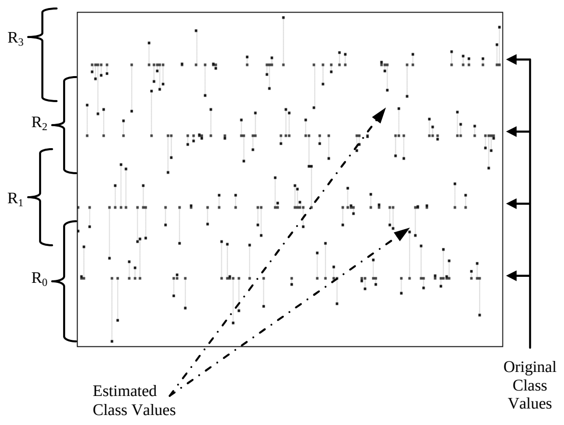

Fig. 3: Visualization of the evaluation of an exemplary classificator.

Unfortunately the presented method for calculating *S* has one disadvantage: Since within the calculation of S_i for a certain class all values of this class have to be compared to all other ones, the computational complexity of this function is quadratic. Especially when it comes to solving bigger problem instances that contain several thousand training samples, this could lead to the fact that the execution of a GA working with a population containing some hundred solution candidates would take much too long if run under normal circumstances (i.e., using a standard PC). Thus, in such cases the set of training samples should be reduced (without omitting the really relevant ones) or the models should be evaluated without considering and calculating *S*.

Fig. 3 visualizes the evaluation of an exemplary classificator. Each test sample object is evaluated and its estimated class value compared to the original one; the resulting class value ranges of the given classes (R_0, R_1, R_2 and R_3) are also shown.

5. ADOPTION OF ROC-CURVES FOR MULTICLASS CLASSIFICATION PROBLEMS

Receiver Operating Characteristic (ROC) curves have long been used in signal detection theory and are able to visualize the quality of a classification hypothesis; good explanations of ROC curves and how the area under them have to be interpreted are for instance given in [6] and [2]. For instances, ROC curves are often used to judge the discrimination ability of classificators or classification methods, respectively.

ROC curves for classification models are calculated as follows: For each possible threshold value discriminating two given classes (e.g., 0 and 1, ‘true’ and ‘false’ or ‘normal’ and ‘diseased’), the numbers of true and false classifications for one of the classes are calculated. For example, if the two classes ‘true’ and ‘false’ are to be discriminated using a given classificator, each possible threshold is tested and the true positives (TP) and the false positives (FP) are counted. Each pair of TP and FP values produces a point of the ROC curve; examples are graphically shown in Fig. 4. Slightly different versions are also often used, for example the Positive Predictive Value (= TP / (TP + FP)) or the Negative Predictive Value (= TN / (FN + TN)) could be displayed instead.

The most common quantitative index describing an ROC curve is the area under it. The bigger the area under a ROC curve is, the better the discriminator model is; if the two classes can be ideally separated, the ROC curve goes through the upper left corner and thus, the area under it is maximally big.

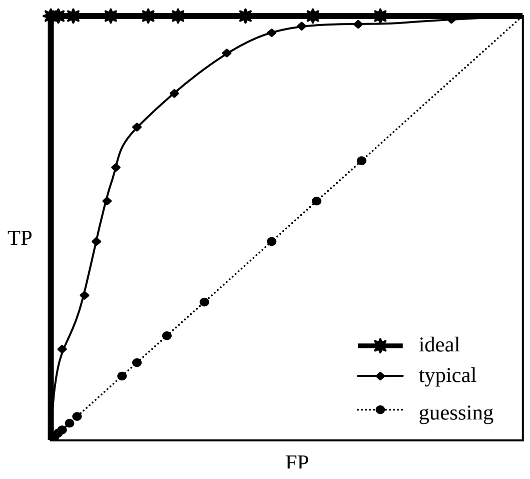

Fig. 4: Examples of ROC curves for classificators discriminating two classes ‘true’ and ‘false’

In fact, the following difficulty arises when evaluating classificators for multiclass problems: For each class c_i optimal thresholds between c_{i-1} and c_i on the one hand and between c_i and c_{i+1} on the other hand have to be found. So we have developed the following adapted version of ROC curves:

For each possible pair of thresholds, the number of correct classifications CC_i with respect to c_i is calculated as well as the number of false classifications FC_i. If an object belonging to c_i is classified as c_i or an object not belonging to c_i is not classified as c_i, then we call this a correct case and increase CC_i; if an

object belonging to c_i is not classified as c_i or an object not belonging to c_i is classified as c_i, then we call this an incorrect case and therefore increase FC_i.
If the CC_i and FC_i values for every possible pair of thresholds are drawn for a class c_i, the resulting graphic does not show a single curve as in the case of ROC curves but rather a set of curves; the better the classificator is, the bigger the area under this new set of adapted ROC curves becomes (i.e., the nearer they come towards the upper left corner of the diagram).

As already mentioned in the previous chapter, the separability value S_i for class i is strongly related to the area under the corresponding ROC curve.

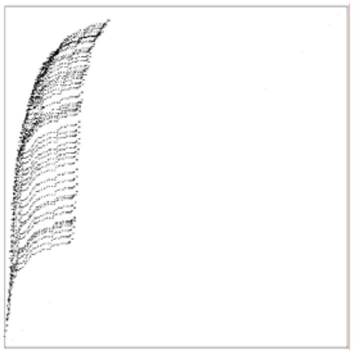

Fig. 5: An exemplary adapted version of ROC curves for one class, produced evaluating a classificator model for an artificially generated multiclass classification example.

6. RESULTS

Empirical studies with various problem instances are possibly the most effective way to analyze the potential of heuristic optimization searches like evolutionary algorithms. Amongst other benchmark problem instances, the GP-based multiclass classification method was tested with the *Thyroid* data set and the *Wisconsin Breast Cancer* data set, benchmark data sets which are included in the UCI Machine Learning Repository[1].

The *Thyroid* data set represents medical measurements which were recorded while investigating patients potentially suffering from hypotiroidism. A detailed description of the problem can be found on the KEEL homepage[2]. In short, the task is to determine whether a patient is hypothyroid or not. Three classes are formed: normal (not hypothyroid), hyperfunction and subnormal functioning; a good classifier has to be significantly better than 92% simply because 92 percent of the patients are not hyperthyroid. In total, the data set contains 7200 samples. For testing our GP-based classification method we have used 6000 samples as training samples available for the GP algorithm and tested the model which was automatically produced by the algorithm with the remaining 1200 samples. The algorithm which was used for all test series was the SASEGASA [1], a new generic evolutionary algorithm for retarding the unwanted effects of premature convergence by combining processing and self-adaptive steering of selection-pressure. The Figures 6 to 8 show the adapted version of ROC curves for the three *Thyroid* classes (based on the evaluation of the whole data set).

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

[2] http://sci2s.ugr.es/keel-dataset/problemas/clasificacion/thyroid.php

Fig. 6: adapted version of ROC curves for Class 0 of the *Thyroid* data set $S_0 = 0.9922$

Fig. 7: adapted version of ROC curves for Class 2 of the *Thyroid* data set $S_2 = 0.9958$

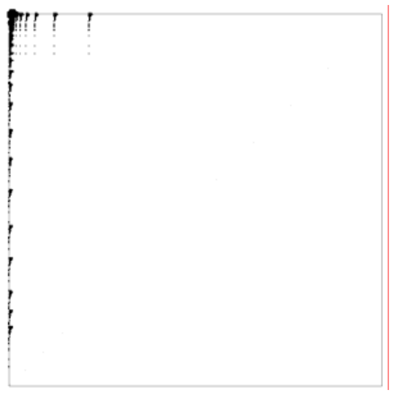

Fig. 8: adapted version of ROC curves for Class 1 of the *Thyroid* dataset (Separability value $S_1 = 0.9987$)

	Evaluation			Prognosis		
	+	-	CC_{tot}	+	-	IC_{tot}
+	315 (69.4%)	2 (0.4%)		51 (63.8%)	0 (0.0%)	
-	6 (1.3%)	131 (28.9%)		2 (2.5%)	27 (33.75%)	
			98.2%			97.5%

Tab. 1: Overview of correct (CC_{tot}) and incorrect classifications (IC_{tot}) for the best threshold between the classes “0” and “1” of the *Thyroid* data set.

	Evaluation			Prognosis		
	+	-	CC_{tot}	+	-	IC_{tot}
+	5519 (94.1%)	28 (0.5%)		1108 (63.8%)	11 (0.9%)	
-	0 (0.0%)	317 (5.4%)		0 (0.0%)	51 (4.4%)	
			99.5%			99.1%

Tab. 2: Overview of correct (CC_{tot}) and incorrect classifications (IC_{tot}) for the best threshold between the classes “1” and “2” of the *Thyroid* data set.

The Tables 1 and 2 give an overview of the quality of the identified classifier: The classes could be separated with an average correctness of 98.88% when applied to the training samples and with an average correctness of 98.28% when applied to the remaining test samples.

The *Wisconsin* data set represents medical measurements which were recorded while investigating patients potentially suffering

from breast cancer. A detailed description of the problem can be found on the KEEL homepage[3]. In short, the task is to determine whether a patient suffers from breast cancer or not; two classes are therefore formed: Suffering from cancer (true) or not (false).
In total, the *Wisconsin* data set contains 682 samples (and is still growing since new measurements are included as soon as they are published). For testing our GP-based classification method we have used 500 samples as training samples available for the GP algorithm and tested the model which was automatically produced by the algorithm with the remaining 182 samples.
The Figures 9 and 10 show the adapted version of ROC curves for the two *Wisconsin* classes (based on the evaluation of the whole data set). Table 3 gives an overview of the quality of the identified classifier: The classes could be separated with 97.8 % correctness when applied to the training samples as well as when applied to the remaining test samples.

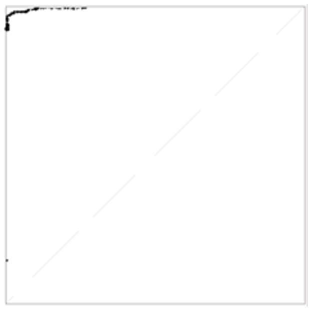
Fig. 9: adapted version of ROC curves for Class 0 of the *Wisconsin* data set $S_0 = 0.99$

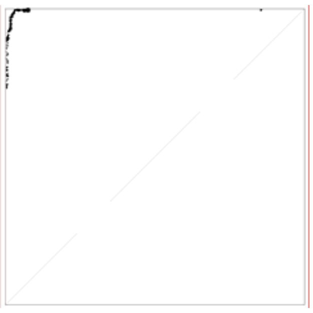
Fig. 10: adapted version of ROC curves for Class 1 of the *Wisconsin* data set $S_1 = 0.99$

	Evaluation			Prognosis		
	1	0	CC_{tot}	1	0	IC_{tot}
1	198 (39.5%)	0 (0.0%)		40 (22.0%)	1 (0.5%)	
0	11 (2.2%)	292 (58.3%)		3 (1.6%)	138 (75.8%)	
			97.8%			97.8%

Tab. 3: Overview of correct (CC_{tot}) and incorrect classifications (IC_{tot}) for the best threshold between the classes "0" and "1" of the *Wisconsin* data set.

The interested reader can find further detailed analysis of the results found for this benchmark data set and also the results we achieved for other benchmark problems in the results section of the HeuristicLab homepage[4].

7. CONCLUSIONS

In this paper we have introduced a genetic programming based method that is able to solve nontrivial real-world multiclass classification problems. A GP approach for identifying nonlinear model structures for mechatronical systems has been further developed and refined so that the basic methods can be used for attacking classification problems yielding very satisfying results. Especially new hybrid variants of genetic algorithms supplementing standard genetic algorithms with artificial self organizing aspects for overcoming some of the fundamental problems of GAs and GP have been applied quite successfully. On the basis of recent test results the authors strongly believe that further refinement of the available operators as well as the design of new operators for this kind of optimization problems will make it possible to find even better results in the field of data mining and classification.

[3] `http://sci2s.ugr.es/keel-dataset/problemas/clasificacion/wisconsin.php`
[4] `www.heuristiclab.com`

8. REFERENCES

[1] M. Affenzeller and S. Wagner, "SASEGASA: A New Generic Parallel Evolutionary Algorithm for Achieving Highest Quality Results", **Journal of Heuristics – Special Issue on New Advances on Parallel Meta-Heuristics for Complex Problems**, Vol. 10, Kluwer Academic Publishers, 2004, pp. 239-263.
[2] A. Bradley, "The Use of the Area under the ROC curve in the Evaluation of Machine Learning Algorithms", **Pattern Recognition**, Vol. 30, 1997, pp. 1145-1159.
[3] U.M. Fayyad, G. Piatetsky-Shapiro and P. Smyth, "From Data Mining to Knowledge Discovery: An Overview", **Advances in Knowledge Discovery and Data Mining**, AAAI Press / MIT Press, 1996.
[4] D.E. Goldberg, **Genetic Algorithms in Search, Optimization and Machine Learning**, Addison Wesley Longman, 1989.
[5] G. Gray et al., "Nonlinear Model Structure Identification Using Genetic Programming", **Control Engineering Practice 6**, 1998.
[6] J. Hanley and B. McNeil, "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve", **Radiology**, Vol. 143, 1982, pp. 29-36.
[7] J.H. Holland, **Adaption in Natural and Artificial Systems**, 1st MIT Press ed., 1992.
[8] J. Koza, **Genetic Programming: On the Programming of Computers by means of Natural Selection**, Cambridge, Mass: The MIT Press, 1992.
[9] W. Langdon and R. Poli, **Foundations of Genetic Programming**, New York: Springer Verlag, 2002.
[10] Z. Michalewicz, **Genetic Algorithms + Data Structures = Evolution Programs**, 3rd ed., Berlin Heidelberg New York: Springer Verlag, 1996.
[11] T.M. Mitchell, **Machine Learning**, McGraw-Hill, 2000.
[12] S. Wagner and M. Affenzeller, "HeuristicLab: A Generic and Extensible Optimization Environment", **Proceedings of the International Conference on Adaptive and Natural Computing Algorithms (ICANNGA)**, 2005.
[13] S. Winkler, M. Affenzeller and S. Wagner, "Identifying Nonlinear Model Structures Using Genetic Programming Techniques", **Cybernetics and Systems 2004**, Austrian Society for Cybernetic Studies, 2004, pp. 689-694.
[14] S. Winkler, M. Affenzeller and S. Wagner, "New Methods for the Identification of Nonlinear Model Structures Based Upon Genetic Programming Techniques", **Proceedings of the 15th International Conference on Systems Science**, Vol. 1, Oficyna Wydawnicza Politechniki Wroclawskiej, 2004, pp. 386-393.
[15] S. Winkler, **Identification of Nonlinear Model Structures by Genetic Programming Techniques**, diploma thesis, Institut für Systemtheorie und Simulation, Technisch Naturwissenschaftliche Fakultät der Johannes Kepler Universität, Linz, Austria, 2004.