Using Enhanced Genetic Programming Techniques for Evolving Classifiers in the Context of Medical Diagnosis - An Empirical Study

To be presented in the
GECCO Workshop on Medical Applications of Genetic and Evolutionary Computation (MedGEC)

Xxxxxxxx Xxxxxxxxxx
Xxxxxxxxxxxx
Xxxxxxxxxxxx
Xxxxxxxxxxxx
Xxxxx@Xxxxxx.xxx

Xxxxxxxx Xxxxxxxxxx
Xxxxxxxxxxxx
Xxxxxxxxxxxx
Xxxxxxxxxxxx
Xxxxx@Xxxxxx.xxx

Xxxxxxxx Xxxxxxxxxx
Xxxxxxxxxxxx
Xxxxxxxxxxxx
Xxxxxxxxxxxx
Xxxxx@Xxxxxx.xxx

ABSTRACT

There are several data based methods in the field of artificial intelligence which are nowadays frequently used for analyzing classification problems in the context of medical applications. As we show in this paper, the application of enhanced evolutionary computation techniques to classification problems has the potential to evolve classifiers of even higher quality than those trained by standard machine learning methods. On the basis of three medical benchmark classification problems, namely the Wisconsin and the Thyroid data sets taken from the UCI repository as well as the Melanoma data set prepared by members of the Department of Dermatology of the Medical University Vienna, we document that the enhanced genetic programming framework presented here is able to produce better results than linear modeling methods, neural networks, kNN classification and also standard genetic programming approaches.

Keywords

Adaptation/Self-Adaptation, Classifier Systems, Genetic Programming, Empirical Study, Medicine

1. INTRODUCTION

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 (as for example 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. The basic steps can be summarized as follows: After training the algorithm using a set of training samples, the algorithm should be able to classify a new, unknown object into one of the predefined classes by analyzing its measured features. Since statistical classification algorithms are supposed to "learn" such functions, we are dealing with a specific subarea of *Machine Learning* and, more generally, *Artificial Intelligence*. There are several approaches that are nowadays used for solving data mining and, more specifically, classification problems. The most common ones are (as for example described in [10]) decision tree learning, instance-based learning, inductive logic programming (such as Prolog, e.g.) and reinforcement learning.

The task of a classification algorithm in the context of patient classification is to find a mathematical description that models underlying classification rules; this means that the diagnosis can be predicted by considering a set of other measured features (for example blood parameter values). Of course, any data based classification algorithm can applied to any given classification problem.

The application of data based machine learning techniques for learning classifiers is one of the major topics not only in computer science in general, but also especially in medical data mining. When it comes to automatically classifying patients into those which are possibly suffering from a certain disease and those which are healthy, one often has to face the problem that special tests (such as blood analysis, e.g.) produce high costs of time and money. This is why one has to seek for hidden relationships between the symptoms of a certain disease and other features which are much easier to

The work described in this paper was done within the Translational Research Project L282 "GP-Based Techniques for the Design of Virtual Sensors" sponsored by the Austrian Science Fund (FWF).

GECCO 2006 Seattle, Washington USA

measure.

In this paper we present a classification algorithm based on genetic programming (GP) [7] using a structure identification framework [17] originally designed for timeseries analysis, described in Section 2.1, in combination with an enhanced, hybrid selection scheme [1] (as explained in Section 2.2). This GP approach has been implemented as a part of the *HeuristicLab* [15], a framework for prototyping and analyzing optimization techniques for which both generic concepts of evolutionary algorithms and many functions to evaluate and analyze them are available.

The results obtained using this method are compared to those achieved by methods that are nowadays standard techniques in machine learning, namely linear regression modeling (Lin), neural networks (NN) and k-nearest-neighbor (kNN) classification; detailed descriptions of these methods, theoretical background and actual variants and applications of these techniques can be found in [9], [4] and [14]. We also compare these results to those achieved using standard GP techniques.

The comparison of the methods mentioned is done on the basis of three medical benchmark classification problems: The Wisconsin and the Thyroid data sets taken from the UCI repository as well as the Melanoma data set prepared by members of the Department of Dermatology of the Medical University Vienna. As we document in Section 3, the enhanced GP method presented here outperforms all other classification methods used in terms of classification accuracy on test data.

2. GP-BASED CLASSIFICATION IN MEDICAL DIAGNOSTICS

2.1 The GP Based Structure Identification Approach

2.1.1 General Remarks

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 (see for instance [17]) that methods of GP are suitable for determining an appropriate mathematical representation of a physical system. The methods implemented for this project have bow been used for developing a GP-based approach for a statistical classification algorithm. This algorithm works on a set of training examples with known properties $[X_1 \dots X_n]$; one of these properties (X_t) has to represent the membership information with respect to the underlying taxonomy. In the context of medical applications, this target property gives the information about the respective patient's diagnosis (as "deseased" vs. "not deseased", e.g.). 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, the GP algorithm works with solution candidates that are tree structure representations of symbolic expressions.

2.1.2 Solution Candidate Representation Using Hybrid Tree Structures

The selection of the library functions is an important part of any GP modeling process [5] because this library should be able to represent a wide range of systems; Table 1 gives an overview of the function set as well as the terminal nodes used for the classification experiments documented in this paper. As the reader can see in Table 1, mathematical functions and terminal nodes are used as well as boolean operators for building complex arithmetic expressions. There are in fact no structural restrictions for the use of boolean blocks in formulae; of course, [Then/Else] and boolean expressions have to be connected to [IF] nodes, but there are no other restrictions regarding the use of boolean blocks within mathematical expressions. Thus, the concept of decision trees is included in this approach together with the standard structure identification concept that tries to evolve nonlinear mathematical expressions. An example showing the structure tree representation of an combined formula including arithmetic as well as logical functions is displayed in Figure 1.

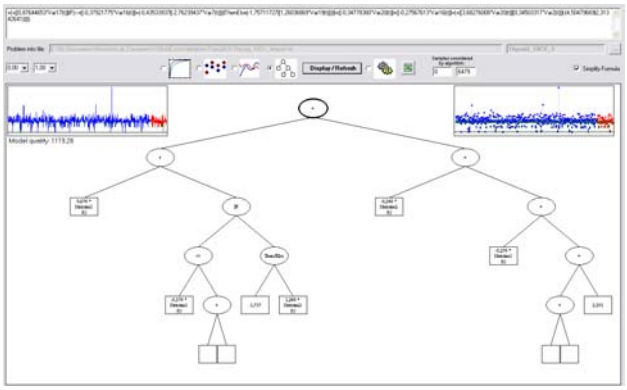

Figure 1: Hybrid Tree Structure.

2.1.3 Evaluation of Classification Models

There are several possible functions that can serve as fitness functions within the GP process. For example, the ratio of misclassifications (using optimal thresholds) or the area under the corresponding ROC curves ([18], [3]) could be used. Still, we have decided to use a variant of the sum of squared errors function for estimation the quality of a classification model. There is one major difference to the standard implementation of this function: The errors of predicted values that are lower than the lowest class value or greater than the greatest class value do not have a quadratic, but only linear contribution to the fitness value. In other words: Given N samples with original classifications o_i divided into n classes $c_1, ..., c_n$ (with c_1 being the lowest and c_n the greatest class value), the fitness value F of a classification model producing the estimated classification values

Table 1: Nodes set for GP based classification.

Functions		
Name	**Arity**	**Description**
$+$	2	Addition
-	2	Subtraction
*	2	Multiplication
/	2	Division
E^x	1	Exponential Function
IF	2	If [Arg1] then return [Then] branch, otherwise return [Else] brach
Then/Else	2	Combined Node for [Then] and [Else] branch
$\leq, \geq$	2	Less or equal, greater or equal
&&, ||	2	Logical AND, logical OR
Terminal nodes		
$[c * Var_X]$		Value of attribute X multiplied with coefficient c
D		A constant double value D

e_i is evaluated as follows:

$$\forall_{i\in[1,n]} : (e_i < c_1 \parallel e_i > c_n) \Rightarrow x_i = \mid e_i - o_i \mid, \quad (1)$$

$$(c_1 \leq e_i \leq c_n) \Rightarrow x_i = (e_i - o_i)^2 \quad (2)$$

The reason for this is that values that are greater than the greatest class value or below the lowest value are classified correctly, anyway; using a standard implementation of the squared error function would punish a formula such values more than necessary.

2.1.4 *Finding Appropriate Class Thresholds*

Of course, a mathematical expression alone does not yet define a classification model; thresholds are used for dividing the output into multiple ranges, each corresponding to exactly one class. These regions are defined before starting the training algorithm in static range selection (SRS) [13], which brings along the difficulty of determining the appropriate range boundaries ahead of time. In the classification framework documented in this paper we have therefore used dynamic range selection (DRS) which attempts to overcome this problem by evolving the range thresholds along with the classification models. The thresholds are chosen so that the sum of ratios of misclassifications for all given classes is minimized (on the training data, of course).

2.2 Introduction of an Enhanced Selection Scheme in the GP Process

In the context of our structure identification approach used for solving classification problems, the standard GP process (basically the procedure described in [7]) is extended by an enhanced selection scheme. Instead of using standard implementations of the genetic algorithm as underlying GP algorithm, a new generic evolutionary algorithm, the SASEGASA [1], is applied. This hybrid GA uses an enhanced selection model which is designed to directly control genetic drift within the population by advantageous self-adaptive selection pressure steering. Additionally, this new selection model enables to detect and combat premature convergence which is generally quite a critical issue in GAs. A very essential question about the general performance of GAs and especially GP is, whether or not good parents are able to produce children of comparable or even better fitness. In natural evolution, this is almost always true. For GAs, this property is not so easy to guarantee and for GP it is a matter of principle that many crossover and mutation results cause counterproductive solution candidates.

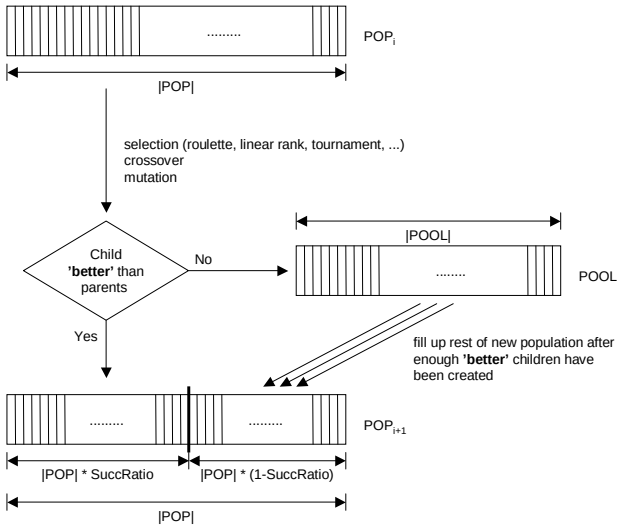

Figure 2: Embedding the new selection principle into a GA or GP.

In order to overcome this drawback, the basic idea of the new selection model which we have called offspring selection (cf. Figure 2) is to consider not only the fitness of the parents in order to produce a child for the ongoing evolutionary process. Additionally, the fitness value of the evenly produced child is compared with the fitness values of its own parents. Basically the child is accepted as a candidate for the further evolutionary process if and only if the reproduction operator was able to produce a child that could outperform the fitness of its own parents. For further details about the more specific parameters and functioning of offspring selection the interested reader is referred to [2].

This strategy guarantees that evolution is continued mainly with crossover results that were able to mix the properties of their parents in an advantageous way which is a very essential aspect for the preservation of essential genetic infor-

mation stored in many individuals (which might not be the fittest in the sense of individual fitness). As elaborate test series have shown (see for example [17] [16] or JoHArticle), the results obtained for various different optimization problems using the SASEGASA were significantly better than those produced by standard GA implementations.

3. EMPIRICAL TEST RESULTS

3.1 Classification Methods Used

For comparing GP based classification with other machine learning methods, the following techniques for training classificators were examined: Genetic programming (enhanced approach as described in Section 2), linear modeling, neural networks and the k-nearest-neighbor method. The benchmark problems analyzed are the data sets *Wisconsin*, *Melanoma* and *Thyroid*.

3.1.1 GP based training of Classifiers

We have used the following parameter settings for our GP test series:

- Population size: 1000
- Mutation rate: 10%
- Selection scheme: SASEGASA selection incorporating offspring selection
- Selection operators: Roulette selection in combination with random selection
- Maximum selection pressure: 555
- Functions set: All functions as described in Section 2.1

In the case of the *Wisconsin* data set, the results retrieved using this enhanced GP (EGP) classification algorithm were compared to those obtained using standard GP techniques. In [13], for example, recent results for several classification benchmark problems are documented; the *Wisconsin* data set is also analyzed using standard GP as well as three other GP based classification variants (POPE-GP, DecMO-GP and DecMOP-GP).

3.1.2 Linear modeling

Given a data collection including m input features storing the information about N samples, a linear model is defined by the vector of coefficients $\theta_{1\dots m}$. For calculating the vector of modeled values e using the given input values matrix $u_{1\dots m}$, these input values are multiplied with the corresponding coefficients and added:

$$e = u_{1\dots m} * \theta \tag{3}$$

The coefficients vector can be computed by simply applying matrix division. For conducting the test series documented here we have used the matrix division function provided by MATLAB:

```
theta = InputValues \ TargetValues;
```

If a constant additive factor is to be included into the model (i.e., the coefficients vector), this command has to be extended:

```
r = size(InputValues,1);
theta = [InputValues ones(r,1)] \ TargetValues;
```

Theoretical background of this approach can be found in [9].

3.1.3 Neural Networks

For training neural network (NN) models, three-layers feed-forward neural networks with one output neuron were created using the Levenberg-Marquardt training method. Theoretical background and details can be found in [11] (Chapter 11, "Neural Networks"), citeMarquardt1963, [8] or [6].
We have trained networks with 5 neurons in the hidden layer (referred to as "NN" in the test series documentation in Section 3.2) as well as networks with 10 hidden neurons (referred to as "NN2" in the test series documentation); the number of iterations of the training process was set to 300. In the context of analyzing the benchmark problems used here, higher numbers of nodes or iterations almost always lead to overfitting (i.e., a better fit on the training data, but worse test results).
The NN training framework used to collect the results reported in this paper is the `NNSYSID20` package, a neural network toolbox for MATLAB implemented by Magnus Nørgaard at the Technical University of Denmark [12].

3.1.4 kNN Classification

Unlike other modeling methods based on linear models, neural networks or GP, k-nearest-neighbor classification works without creating any explicit models. During the training phase, the data are simply collected; when it comes to classifying a new, unknown sample x_new, the sample-wise distance x_new and all other training samples x_train is calculated and the classification is done on the basis of those k training samples (x_{NN}) showing the smallest distances from x_new.

The distance between two samples is calculated as follows: First, all features are normalized by subtracting the respective mean values and dividing the remaining samples by the respective variables' variances. Given a data matrix x including m features storing the information about N samples, the normalized values x_{norm} are calculated as

$$\forall_{i\in[1,m]}\forall_{j\in[1,N]} : x_{norm}(i,j) = \frac{x(i,j) - \frac{1}{N}\sum_{k=1}^{N} x(i,k)}{var(x(i,1\dots N))} \tag{4}$$

where the variance var of a given variable x storing N values is calculated as

$$var(x) = \frac{1}{N-1}\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{5}$$

with $\bar{x}$ denoting the mean value of x.
Then, on the basis of the normalized data, the distance between two samples a and b, $d(a,b)$, is calculated as the mean squared variable-wise distance:

$$d(a,b) = \frac{1}{n}\sum_{i=1}^{n}\left(a_{norm}(i) - b_{norm}(i)\right)^2 \tag{6}$$

where n again is the number of features stored for each sample.

In the context of classification, the numbers of instances (of the k nearest neighbors) are counted for each given class and the algorithm automatically predicts that class that is represented by the highest number of instances (included in x_{NN}). In the test series documented in this paper we have applied weighting to kNN classification: The distance between x_{new} and x_{NN} is relevant for the classification state-

ment, the weight of nearer samples is higher than that of samples that are "further" away from x_new.

There is a lot of literature that can be found for kNN classification; very good explanations and compact overviews of kNN classification (including several possible variants and applications) are for example given in [4] and [14].

3.2 Results

All three data sets were investigated via 10-fold cross-validation. This means that each original data set was divided into 10 disjoint sets of (approximately) equal size. Thus, 10 different pairs of training (90% of the data) and test data sets (10% of the data) can be formed and used for testing the classification algorithm. For the *Wisconsin* and the *Melanoma* data set we give the percentages of correct classifications for each partition and each modeling method tested. In the case of the *Thyroid* data set, the samples are not equally distributed to the three given classes; this is why we give the percentages of correct classifications for the *Thyroid* problem for each class separately.

3.2.1 Wisconsin

The *Wisconsin* data set is a part of the UCI Machine Learning Repositoy; in short, it represents medical measurements which were recorded while investigating patients potentially suffering from breast cancer. The number of features recorded is 9, the file version we have used contains 683 recorded examples (by now, 699 examples are already available since the data base is updated regularily). A detailed description of the problem can be found on the KEEL homepage; the best results published using various classificators (the best ones using problem specific methods achieving up to 98.7% accuracy on the training data and 98.5% on the test data) can be found there, too.

In Table 2 we give the percentages of correct classifications for each class evaluated on training data and on test data, respectively. The following training methods have been used: Linear modeling (*"Lin"*), neural networks (*"NN1"*: 5 hidden nodes, *"NN2"*: 10 hidden nodes), kNN classification (*"kNN"*, $k = 3$) and enhanced GP (*"EGP"*.
An overview of the average percentages of correct classification for each method tested is given in Table 3. Additionally, we have compared the results to those achieved using other GP implementations documented in [13], namely *"Standard GP"*, *"POPE-GP"*, *"DecMO-GP"* and *"DecMOP-GP"*.

3.2.2 Melanoma

The *Melanoma* data set represents medical measurements which were recorded while investigating patients potentially suffering from skin cancer. It contains 1311 examples for which 30 features have been recorded; it has been provided to us by Prof. Michael Binder from the Departement of Dermatology at the Medical University Vienna, Austria.

In Table 4 we give the percentages of correct classifications for each class evaluated on training data and on test data, respectively. The following training methods have been used: Linear modeling (*"Lin"*), neural networks (*"NN1"*: 5 hidden nodes, *"NN2"*: 10 hidden nodes), kNN classification (*"kNN"*, $k = 3$) and enhanced GP (*"EGP"*. An overview of the average percentages of correct classification for each method tested is given in Table 5.

`http://www.ics.uci.edu/~mlearn/`
`http://sci2s.ugr.es/keel-dataset/`

Table 2: Percentual correct classification rates for the classifiers produced for the 10-fold CV partitions of the *Wisconsin* data set.

	Evaluation on training data				
Part.	*Lin*	*NN*	*NN2*	*kNN*	*EGP*
0	97.88	100.00	100.00	-	98.37
1	97.88	99.18	100.00	-	97.56
2	97.56	99.83	100.00	-	99.02
3	98.21	99.83	100.00	-	98.86
4	97.88	100.00	100.00	-	97.06
5	97.56	100.00	100.00	-	98.37
6	97.56	99.34	100.00	-	98.21
7	97.39	99.67	100.00	-	98.05
8	97.88	100.00	100.00	-	97.56
9	97.56	99.34	100.00	-	98.21
avg.	97.74	99.69	99.98	-	98.08
	Evaluation on test data				
Part.	*Lin*	*NN*	*NN2*	*kNN*	*EGP*
0	94.12	92.64	86.76	85.29	95.59
1	98.53	92.75	94.12	100.00	98.53
2	94.12	98.52	89.70	95.59	98.53
3	94.12	94.12	89.70	89.70	94.12
4	95.65	92.75	94.12	95.65	97.14
5	94.12	94.12	92.64	94.12	98.53
6	98.53	97.14	98.53	98.53	95.59
7	100.00	97.14	100.00	97.14	100.00
8	97.14	91.30	92.64	98.53	95.59
9	94.12	95.59	98.53	94.12	97.10
avg.	96.05	94.59	93.70	94.86	97.07

Table 3: Average correct classification rates (given as percentual values) for the classifiers produced for the *Wisconsin* data set.

Algorithm	*Training*	*Test*
Lin	97.74	96.05
NN	99.69	94.59
NN2	99.98	93.70
kNN	-	94.86
StandardGP	98.98	93.82
POPE − GP	99.24	95.08
DecMO − GP	99.40	95.19
DecMOP − GP	99.27	95.60
SASEGASA − GP	98.08	97.07

3.2.3 Thyroid

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. 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.

As already mentioned, the samples of the *Thyroid* data

Table 4: Percentual correct classification rates for the classifiers produced for the 10-fold CV partitions of the *Melanoma* data set.

	Evaluation on training data				
Part.	*Lin*	*NN*	*NN*2	*kNN*	*EGP*
0	93.64	97.20	99.91	-	97.29
1	92.37	99.40	98.13	-	95.51
2	94.06	99.91	99.91	-	96.36
3	93.13	98.30	98.64	-	96.36
4	93.72	96.77	98.55	-	96.27
5	94.06	99.49	97.28	-	96.36
6	94.74	99.66	100.00	-	97.12
7	92.45	98.30	99.91	-	96.19
8	93.22	98.89	98.47	-	95.59
9	93.72	98.38	99.74	-	96.19
avg.	93.51	98.63	99.05	-	96.24
	Evaluation on test data				
Part.	*Lin*	*NN*	*NN*2	*kNN*	*EGP*
0	88.54	92.36	93.12	97.70	95.42
1	90.83	96.94	93.89	96.94	98.47
2	92.36	95.41	90.07	93.12	96.18
3	93.89	93.12	90.07	88.54	93.13
4	89.39	96.21	93.93	87.12	91.60
5	93.89	95.41	96.94	94.65	96.95
6	93.12	90.07	91.60	93.89	93.13
7	96.18	96.94	92.36	96.18	96.18
8	96.94	95.41	96.94	94.65	96.95
9	89.31	91.60	90.07	93.12	96.18
avg.	92.45	94.35	92.90	93.59	95.42

set are not equally distributed to the three given classes; in fact, 166 samples belong to class '1' ('subnormal functioning'), 368 samples are classified as '2' ('hyperfunction'), and the remaining 6666 samples belong to class '3' ('normal, not hypothyroid'). This is why we give the percentages of correct classifications for the *Thyroid* problem for each class separately. For the sake of readability we do not state the results for every partition, every class and every method; in the Tables 6 and 6 we give the percentages of correct classifications for each class in terms of average accuracy, standard deviation, minimum and maximum values for the methods tested on training data and on test data, respectively.
The following training methods have been used: Linear modeling, neural networks ("Variant 1": 5 hidden nodes, "Variant 2": 10 hidden nodes), kNN classification ($k = 5$) and enhanced GP.

4. DISCUSSION

As documented in the Tables 2 until 7 it was able to show that for all three medical benchmark data sets the results achieved using the enhanced GP approach presented in this paper are better than those achieved applying linear regression, neural networks, neighborhood classification or standard genetic programming.

Evaluated on the first data collection, the *Wisconsin* data set, neural networks are able to approximate the given data best showing a almost perfect fit (99.69% and 99.98% correct classifications, respectively) while enhanced GP is only able to produce classifiers with an average 98.09% classification accuracy. But when it comes to testing new, unknown data, the EGP method performs best; while the evaluation of the EGP method yields less than 3% test error, all other methods perform significantly worse with average test errors ranging from 6.18% (standard GP) to 3.95% (linear modeling).

Table 5: Average percentual correct classification rates for the classifiers produced for the *Melanoma* data set.

Algorithm	*TrainingError*	*TestError*
Lin	93.51	92.45
NN	98.63	94.35
*NN*2	99.05	92.90
kNN	-	93.59
SASEGASA − GP	96.24	95.42

Table 6: Summary of training results for the *Thyroid* data set: correct classifications (given as percentual values) for the 10-fold CV partitions.

Evaluation of Linear Model Classifiers			
	*Class*1	*Class*2	*Class*3
Avg	33.60	22.96	99.08
Std.Dev.	1.51	3.22	0.15
Min	31.33	19.22	98.78
Max	35.81	30.47	99.28
Evaluation of NN Classifiers (Var. 1)			
	*Class*1	*Class*2	*Class*3
Avg	91.47	94.15	99.24
Std.Dev.	4.82	4.12	0.27
Min	82.78	84.62	98.82
Max	97.39	99.07	99.63
Evaluation of NN Classifiers (Var. 2)			
	*Class*1	*Class*2	*Class*3
Avg	93.73	96.10	99.50
Std.Dev.	2.56	1.01	0.14
Min	91.21	94.34	99.27
Max	97.76	97.93	99.70
Evaluation of enhanced GP Classifiers			
	*Class*1	*Class*2	*Class*3
Avg	93.72	96.68	99.46
Std.Dev.	3.19	3.42	0.30
Min	90.26	88.17	98.80
Max	98.68	100.00	99.83

The evaluation of the classification method investigated using the *Melanoma* data set shows similar results:
Enhanced GP is surely not the method for approximating given data as well as possible. While neural networks show a training error of less than 1.5%, the training error rate of the EGP method is more than twice as high (3.76%); only linear modeling performs worse (only 93.51% correct classification on training data). The test phase again shows that genetic programming methods are able to perform better than classical machine learning methods: The test error of the EGP method is 4.58% whereas neural networks produce 5.65% and 7.1% false classifications, respectively, and linear regression is also able to classify only 92.45% of the test

Table 7: Summary of test results for the *Thyroid* data set: correct classifications (given as percentual values) for the 10-fold CV partitions.

Evaluation of Linear Model Classifiers			
	*Class*1	*Class*2	*Class*3
Avg	33.39	22.23	99.05
Std.Dev.	12.47	6.87	0.41
Min	11.11	10.26	98.62
Max	56.25	30.00	99.84
Evaluation of NN Classifiers (Var. 1)			
	*Class*1	*Class*2	*Class*3
Avg	84.93	91.01	98.68
Std.Dev.	6.11	6.11	0.83
Min	75.00	83.33	97.45
Max	94.44	100.00	99.55
Evaluation of NN Classifiers (Var. 2)			
	*Class*1	*Class*2	*Class*3
Avg	85.40	92.46	97.91
Std.Dev.	8.28	6.31	2.70
Min	73.68	80.56	90.40
Max	94.44	100.00	99.40
Evaluation of kNN Classifiers			
	*Class*1	*Class*2	*Class*3
Avg	75.00	32.52	99.42
Std.Dev.	14.36	21.60	0.67
Min	52.63	8.88	97.61
Max	88.89	82.05	99.87
Evaluation of enhanced GP Classifiers			
	*Class*1	*Class*2	*Class*3
Avg	88.45	93.09	99.40
Std.Dev.	7.87	6.75	0.43
Min	75.00	80.00	98.65
Max	100.00	100.00	100.00

samples correctly. Only kNN classification (5.65% test error) is able to perform almost as well, but still not quite as well as EGP.

The most challenging data set investigated is the *Thyroid* data set. As can be seen in Table 6, the linear modeling method is not able to produce satisfying classifiers. Neural networks are trained showing rather good fit on the training data: 91.47% (and 93.73%, respectively) of the training samples belonging to class 1 can be reconstructed correctly, 94.15% (96.68%) of class 2 and more than 99% of the training samples classified as "not hyperthyroid". EGP produces classifiers that are quite as good for class 1 and even a little bit better for classes 2 and 3 evaluated on the training data. As documented in Table 7, EGP obviously outperforms the other methods investigated significantly: kNN classification (with $k = 5$), for example, is on the average only able to correctly detect 32.52% of the test samples originally classified as '2', NN classifiers show lower correct classification rates then EGP. EGP yields best classifiers for the classes 1 and 2, only for class '3' kNN performs slightly better (99.42% correct classification opposed to 99.40% achieved by EGP).

5. CONCLUSION

In this paper we have presented an enhanced genetic programming method that was successfully used for investigating machine learning problems in the context of medical classification. The approach works with hybrid formula structures combining logical expressions (as used for example in decision trees) and classical mathematical functions; the enhanced selection scheme originally successfully applied for solving combinatorial optimization problems using genetic algorithms was also applied yielding high quality results.

As documented in the detailed test results summary, this genetic programming based classification approach significantly outperforms classical machine learning algorithms frequently used for solving classification problems, namely linear regression, neural networks and neighborhood based classification.

The results achieved for the medical data collections are very satisfying and make the authors believe that an application in a real-world framework in the context of medical data analysis is possible using the techniques presented here.

6. 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*, 10:239–263, 2004.

[2] M. Affenzeller and S. Wagner. Offspring selection: A new self-adaptive selection scheme for genetic algorithms. *Adaptive and Natural Computing Algorithms*, pages 218–221, 2005.

[3] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[4] R. O. Duda, P. E. Heart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2000.

[5] G. G. et al. Nonlinear model structure identification using genetic programming. *Control Engineering Practice*, 6, 1998.

[6] P. R. Gill, W. Murray, and M. H. Wright. The levenberg-marquardt method. *Practical Optimization*, pages 136–137, 1981.

[7] J. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection*. The MIT Press, Cambridge, Mass, 1992.

[8] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[9] L. Ljung. *System Identification*. Prentice Hall, 1999.

[10] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 2000.

[11] O. Nelles. *Nonlinear System Identification*. Springer Verlag, Berlin Heidelberg New York, 2001.

[12] M. Norgaard. Neural network based system identification toolbox. Technical Report Technical Report 00-E-891, Technical University of Denmark, 2000.

[13] D. Parrott, X. Li, and V. Ciesielski. Multi-objective techniques in genetic programming for evolving classifiers. *Proceedings of the 2005 Congress on Evolutionary Computation (CEC '05)*, pages 183–190, 2005.

[14] S. Russell and P. Norvig. *Artificial Intelligence: A Moden Approach*. Prentice Hall, 2000.

[15] S. Wagner and M. Affenzeller. Heuristiclab: A generic and extensible optimization environment. *Adaptive and Natural Computing Algorithms*, pages 538–541, 2005.

[16] S. Winkler, M. Affenzeller, and S. Wagner. A genetic programming based tool for supporting bioinformatical classification problems. *Proceedings of the FH Science Day 2005*, pages 3–10, 2005.

[17] S. Winkler, M. Affenzeller, and S. Wagner. New methods for the identification of nonlinear model structures based upon genetic programming techniques. *Journal of Systems Science*, 31(1):5–13, 2005.

[18] M. H. Zweig. Receiver-operating characteristic (ROC) plots: A fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, 39:561–577, 1993.