Information Sciences 222 (2013) 805–823

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis

Jose G. Moreno-Torres [a,*], Xavier Llorà [b], David E. Goldberg [c], Rohit Bhargava [d]

[a] *Department of Computer Science and Artificial Intelligence, Universidad de Granada, 18071 Granada, Spain*
[b] *National Center for Supercomputing Applications (NCSA), University of Illinois at Urbana-Champaign 1205 W. Clark Street, Urbana, Illinois, USA*
[c] *Illinois Genetic Algorithms Laboratory (IlliGAL) University of Illinois at Urbana-Champaign 104 S. Mathews Ave, Urbana, Illinois, USA*
[d] *Department of Bioengineering, University of Illinois at Urbana-Champaign 405 N. Mathews Ave, Urbana, Illinois, USA*

A R T I C L E   I N F O

Article history:
Available online 6 October 2010

Keywords:
Genetic programming
Feature extraction
Fractures between data
Biological data
Cancer diagnosis
Different laboratories

A B S T R A C T

There is an underlying assumption on most model building processes: given a learned classifier, it should be usable to explain unseen data from the same given problem. Despite this seemingly reasonable assumption, when dealing with biological data it tends to fail; where classifiers built out of data generated using the same protocols in two different laboratories can lead to two different, non-interchangeable, classifiers. There are usually too many uncontrollable variables in the process of generating data in the lab and biological variations, and small differences can lead to very different data distributions, with a fracture between data.

This paper presents a genetics-based machine learning approach that performs feature extraction on data from a lab to help increase the classification performance of an existing classifier that was built using the data from a different laboratory which uses the same protocols, while learning about the shape of the fractures between data that motivated the bad behavior.

The experimental analysis over benchmark problems together with a real-world problem on prostate cancer diagnosis show the good behavior of the proposed algorithm.

1. Introduction

The assumption that a properly trained classifier will be able to predict the behavior of unseen data from the same problem is at the core of any automatic classification process. However, this hypothesis tends to prove unreliable when dealing with biological (or other experimental sciences) data, especially when such data is provided by more than one laboratory, even if they are following the same protocols to obtain it.

The specific problem this paper attempts to solve is the following: we have data from one laboratory (dataset A), and derive a classifier from it that can predict its category accurately. We are then presented with data from a second laboratory (dataset B). This second dataset is not accurately predicted by the classifier we had previously built due to a fracture between the data of both laboratories. We intend to find a transformation of dataset B (dataset S) where the classifier works.

Evolutionary computing, as introduced by Holland [29]; is based on the idea of the survival of the fittest, evoked by the natural evolutionary process. In genetic algorithms (GAs) [22], solutions (genes) are more likely to reproduce the fitter

* Corresponding author. Tel.: +34588998183.
E-mail addresses: jose.garcia.mt@decsai.ugr.es (J.G. Moreno-Torres), xllora@illinois.edu (X. Llorà), deg@illinois.edu (D.E. Goldberg), rbx@uiuc.edu (R. Bhargava).

0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.09.018

they are, and random sporadic mutations help maintain population diversity. Genetic Programming (GP) [35] is a development of those techniques, and follows a similar pattern to evolve tree-shaped solutions using variable-length chromosomes.

Feature extraction, as defined by Wyse et al. [58], 'consists of the extraction a set of new features from the original features through some functional mapping'. Our approach to the problem can be seen as feature extraction, since we build a new set of features which are functions of the old ones. However, we have a different goal than that of classical feature extraction, since our intention is to fit a dataset to an already existing classifier, not to improve the performance of a future one.

In this work, we intend to demonstrate the use of GP-based feature extraction to unveil transformations in order to improve the accuracy of a previously built classifier, by performing feature extraction on a dataset where said classifier should, in principle, work; but where it does not perform accurately enough. We test our algorithm first on artificially-built problems (where we apply ad hoc transformations to datasets from which a classifier has been built, and use the dataset resulting from those transformations as our problem dataset); and then on a real-world application where biological data from two different medical laboratories regarding prostate cancer diagnosis are used as datasets A and B.

Even though the method proposed in this paper does not attempt to reduce the number of features or instances in the dataset, it can still be regarded as a form of data reduction because it unifies the data distributions of two datasets; which results in the capability of applying the same classifier to both of them, instead of needing two different classifiers, one for each dataset.

The remainder of this paper is organized as follows: in Section 2, some preliminaries about the techniques used and some approaches to similar problems in the literature are presented. Section 3 details the real-world biological problem that motivates this paper. Section 4 has a description of the proposed algorithm GP-RFD; and Section 5 includes the experimental setup, along with the results obtained, and an analysis. Finally, in Section 6 some concluding remarks are made.

2. Preliminaries

This section is divided in the following way: in Subsection 2.1 we introduce the notation that has been used in this paper. Then we include an introduction to GP in Subsection 2.2, a brief summary of what has been done in feature extraction in Subsection 2.3, and a short review of the different approaches we found in the specialized literature on the use of GP for feature extraction in Subsection 2.4. We conclude mentioning some works related to the finding and repair of fractures between data in Subsection 2.5.

2.1. Notation

A classification problem is considered with:

- A set of input variables $X = \{x_i / i = 1, \ldots, n_v\}$, where n_v is the number of features (attributes) of the problem.
- A set of values for the target variable (class) $C = \{C^j / j = \{1, \ldots, n_c\}\}$, where n_c is the number of different values for the class variable.
- A set of examples $E = \{e^h = (e^h_1, \ldots, e^h_{n_v}, C^h) / h = 1, \ldots, n_e\}$, where C^h is the class label for the sample e^h, and n_e is the number of examples.

When describing the problem, we mention datasets A, B and S. They correspond to:

- A: the original dataset that was used to build the classifier.
- B: the problem dataset. The classifier is not accurate on this dataset, and that is what the proposed algorithm attempts to solve.
- S: the solution dataset, result of applying the evolved transformation to the samples in dataset B. The goal is to have the classifier performance be as high as possible on this dataset.

When performing experiments and obtaining the evolved expressions, we use the following notation: when artificially creating a dataset B by means of a fabricated transformation over dataset A, we have $B = \{b_i / i = 1, \ldots, n_v\}$ be the attributes in dataset B and $A = \{a_i / i = 1, \ldots, n_v\}$ be the ones from dataset A. In appendix A, we show the learned transformations for the prostate cancer problem. The attributes shown are those corresponding to dataset S, and are represented as $S = \{s_i / i = 1, \ldots, n_v\}$.

2.2. Genetic programming

A GA [22] is a stochastic optimization technique inspired by nature's development of useful characters. It is based on the idea of survival of the fittest [12] in the following way: given a population of possible solutions to a problem (represented by chromosomes), there is some selection procedure that favors the fitter ones (i.e., the ones that provide a higher-quality solution); and the selected chromosomes get an opportunity to pass down their genetic material to the next generation via some crossover operator; which usually builds new individuals from the combination of old ones. In some variations of the algorithm, random mutations are sporadically introduced to help maintain biological diversity in the population.

GP, as proposed by John Koza in 1992 [35], uses a selectorecombinative schema where the solutions are represented by trees; which are encoded as variable-length chromosomes. It was originally designed to automatically develop programs, but it has been used for multiple purposes due to its high expressive power and flexibility. In the words of Poli and Langdon [48], 'GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done. Using ideas from natural evolution, GP starts from an ooze of random computer programs, and progressively refines them through processes of mutation and sexual recombination, until solutions emerge. This is all done without the user having to know or specify the form or structure of solutions in advance. GP has generated a plethora of human-competitive results and applications, including novel scientific discoveries and patentable inventions'.

There are a few details about GP that make it different from standard GAs:

- Crossover: the most commonly used operator is one-point crossover, which is analogous to the GA classical one, but where subtrees instead of a specific gene signal where the cut is made.
- Even though mutation was used in the early literature regarding the evolution of programs (see [7,11,17]) Koza chose not to use it ([35,36]), as he wished to demonstrate that mutation was not necessary. This has significantly influenced the field, and mutation was often omitted from GP runs. However, mutation has proved useful since then (see [5,44], for example); and its use is widely spread nowadays. Multiple different mutation operators have been proposed in the literature [46].
- Treatment of constants: the discovery of constants is one of the hardest issues in GP. Koza proposed a solution called Ephemeral Random Constant (ERC), which uses a fixed terminal (e) to represent a constant. The first time one of such constants is evaluated, it gets assigned a random value. From there on, it retains that value throughout the whole run. A number of alternatives have been proposed in the literature [15,51], but ERC remains the most used one.
- Automatically defined functions: ADFs were also first proposed by Koza [36]. The idea is to permit each individual to evolve more than one tree simultaneously; having the extra trees work as primitives that can be called from the main one.

GP has been applied often to classification [14]. Among the latest advances in the field, we would like to mention those dedicated to high dimensional problems [37,6], variations in population size [33,34], and applications to other related fields [60,3].

2.3. Feature extraction

Feature extraction creates new features as functional mappings of the old ones. It has been used both as a form of preprocessing, which is the approach we use in this paper, and also embedded with a learning process in wrapper techniques. An early proposer of such a term was probably Wyse in 1980, in a paper about intrinsic dimensionality estimation [58]. There are multiple techniques that have been applied to feature extraction throughout the years, ranging from principal component analysis to support vector machines to GAs (see [30,47,45], respectively, for some examples).

Among the foundations papers in the literature, Liu's book in 1998 [40] is one of the earlier compilations of the field. As a result of a workshop held in 2003 [25], Guyon and Elisseeff published a book with an important treatment of the foundations [26].

2.4. Genetic programming-based feature extraction

GP has been used extensively to optimize feature extraction and selection tasks. One of the first contributions in this line was the one published by Tackett in 1993 [55], who applied GP to feature discovery and image discrimination tasks.

We can consider two main branches in the philosophy of GP-based feature extraction:

On one hand, we have the proposals that focus only on the feature extraction procedure, of which there are multiple examples: Sherrah et al. [52] presented in 1997 the evolutionary pre-processor (EPrep), which searches for an optimal feature extractor by minimizing the misclassification error over three randomly selected classifiers. Kotani et al.'s work from 1999 [32] determined the optimal polynomial combinations of raw features to pass to a k-nearest neighbor classifier. In 2001, Bot [8] evolved transformed features, one-at-a-time, again for a k-NN classifier, utilizing each new feature only if it improved the overall classification performance. Zhang and Rockett, in 2006, [63] used multiobjective GP to learn optimal feature extraction in order to fold the high-dimensional pattern vector to a one-dimensional decision space where the classification would be trivial. Lastly, also in 2006, Guo and Nandi [24] optimized a modified Fisher discriminant using GP, and then Zhang et al. extended their work by using a multiobjective approach to prevent tree bloat [64], and applied a similar method to spam filtering [62].

On the other hand, some authors have chosen to evolve a full classifier with an embedded feature extraction step. As an example, Harris [28] proposed in 1997 a co-evolutionary strategy involving the simultaneous evolution of the feature extraction procedure along with a classifier. More recently, Smith and Bull [54] developed a hybrid feature construction and selection method using GP together with a GA. FLGP, by Yin et al. [39] is yet another example, where 'new features extracted by certain layer are used to be the training set of next layer's populations'.

2.5. Finding and repairing fractures between data

Throughout the literature there have been a number proposals to quantify the amount of dataset shift (in other words, the size of the fracture in the data). This shift is usually due to time passing (the data comes from the same source at a latter time), but it can also be due to the data being originated by different sources, as is the case in this paper. Some of the most relevant works in the field are: Wang et al. [56], where the authors present the idea of correspondence tracing. They propose an algorithm for the discovering of changes of classification characteristics, which is based on the comparison between two rule-based classifiers, one built from each dataset. Yang et al. [59] presented in 2008 the idea of conceptual equivalence as a method for contrast mining, which consists of the discovery of discrepancies between datasets. Lately, it is important to mention the work by Cieslak and Chawla [10], which presents a statistical framework to analyze changes in data distribution resulting in fractures between the data.

A different approach to fixing data fractures relies on the adaptation of the classifier. Quiñonero-Candela et al. [49] edited a very interesting book on the topic, including several specific proposals to repair fractures between data (what they call dataset shift). There are two main differences between the usual proposals in the literature and this contribution: first, they are most often based on altering the classifier, while we propose keeping it intact and transforming the data. Second, most authors focus on covariate shift, a specific kind of data fracture, but the method we propose here is more general and can tackle any kind of shift.

3. Case study: prostate cancer diagnosis

This section begins with an introduction to the importance of the problem in Subsection 3.1. The diagnostic procedure is summarized in Subsection 3.2, and the reason to apply GP-RFD to this problem is shown in Subsection 3.3. Finally, the preprocessing the data went through is presented in Subsection 3.4.

3.1. Motivation

Prostate cancer is the most common non-skin malignancy in the western world. The American Cancer Society estimated 192,280 new cases and 27,360 deaths related to prostate cancer in 2009 [2]. Recognizing the public health implications of this disease, men are actively screened through digital rectal examinations and/or serum prostate specific antigen (PSA) level testing. If these screening tests are suspicious, prostate tissue is extracted, or biopsied, from the patient and examined for structural alterations. Due to imperfect screening technologies and repeated examinations, it is estimated that more than one million people undergo biopsies in the US alone.

3.2. Diagnostic procedure

Biopsy, followed by manual examination under a microscope is the primary means to definitively diagnose prostate cancer as well as most internal cancers in the human body. Pathologists are trained to recognize patterns of disease in the architecture of tissue, local structural morphology and alterations in cell size and shape. Specific patterns of specific cell types distinguish cancerous and non-cancerous tissues. Hence, the primary task of the pathologist examining tissue for cancer is to locate foci of the cell of interest and examine them for alterations indicative of disease. A detailed explanation of the procedure is beyond the scope of this paper and can be found elsewhere [38,43,42].

Operator fatigue is well-documented and guidelines limit the workload and rate of examination of samples by a single operator (examination speed and throughput). Importantly, inter- and intra-pathologist variation complicates decision making. For this reason, it would be extremely interesting to have an accurate automatic classifier to help reduce the load on the pathologists. This was partially achieved in [43], but some issues remain open.

3.3. The generalization problem

Llorà et al. [43] successfully applied a genetics-based approach to the development of a classifier that obtained human-competitive results based on FTIR data. However, the classifier built from the data obtained from one laboratory proved remarkably inaccurate when applied to classify data from a different hospital. Since all the experimental procedure was identical; using the same machine, measuring and post-processing; and having the exact same lab protocols, both for tissue extraction and staining; there was no factor that could explain this discrepancy.

What we attempt to do with this work is develop an algorithm that can evolve a transformation over the data from the second laboratory, creating a new dataset where the classifier built from the first lab is as accurate as possible. This evolved transformation would also provide valuable information, since it would allow the scientists processing the tissues analyze the differences between their results and those of other hospitals.

3.4. Pre-processing of the data

The biological data obtained from the laboratories has an enormous size (in the range of 14 GB of storage per sample); and parallel computing was needed to achieve better-than-human results. For this reason, feature selection was performed on the dataset obtained by FTIR. It was done by applying an evaluation of pairwise error and incremental increase in classification accuracy for every class, resulting in a subset of 93 attributes. This reduced dataset provided enough information for classifier performance to be rather satisfactory: a simple C4.5 classifier achieved $\sim$95% accuracy on the data from the first lab, but only $\sim$80% on the second one. The dataset consists of 789 samples from one laboratory and 665 from the other one. These samples represent 0.01% of the total data available for each data set, which were selected applying stratified sampling without replacement. A detailed description of the data pre-processing procedure can be found in [16].

4. A proposal for GP-based feature extraction for the repairing of fractures between data (GP-RFD)

This section is presented in the following way: first, a justification for the choice of GP is included. Subsection 4.1 details how the solutions are represented, then the fitness evaluation procedure and the genetic operators are introduced in Subsections 4.2 and 4.3 respectively. Then, the parameter choices are explained in Subsection 4.4, while the function set is in Subsection 4.5. Finally, the execution flow of the whole procedure is shown in Subsection 4.6.

The problem we are attempting to solve is the design of a method that can create a transformation from a dataset (dataset B) where a classification model is not accurate enough into a new one where it is (dataset S). Said classifier is kept unchanged throughout the process.

We decided to use GP to solve the problem for a number of reasons: first, it is well suited to evolve arbitrary expressions because its chromosomes are trees. This is useful in our case because we want to have the maximum possible flexibility in terms of the functional expressions that can be present in the feature extraction procedure. Second, GP provides highly-interpretable solutions. This is an advantage because our goal is not only to have a new dataset where the classifier works, but also to analyze what was the problem in the first dataset.

The specific decisions to be made once GP was chosen as the technique to apply are how to represent the solutions, what terminals and operators to choose, how to calculate the fitness of an individual and which evolutionary parameters (population size, number of generations, selection and mutation rates, etc.) are appropriate for each specific problem. To clarify some of the points, let us have a binary 2-dimensional problem as an example, and let us use a function set composed of $\{+, -, *, \div\}$.

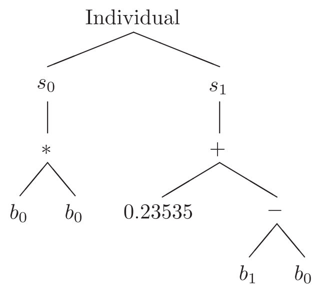
4.1. Solutions representation: context-free grammar

The representation issue was solved by extending GP to evolve more than one tree per solution. Each individual is composed by n trees, where $n = n_v$, the number of attributes present in the dataset (we are trying to develop a new dataset with the same number of attributes as the old one). In the tree structure, the leaves are either constants (we use the Ephemeral Random Constant approach) or attributes from the original dataset. The intermediate nodes are functions from the function set, which is specific to each problem.

The attributes on the transformed dataset are represented by algebraic expressions. These expressions are generated according to the rules of a context-free grammar which allows the absence of some of the functions or terminals. The grammar corresponding to the example problem would look like this:

$$\begin{aligned}
&Start \rightarrow Tree\ Tree\\
&Tree \rightarrow Node\\
&Node \rightarrow Node\ Operator\ Node\\
&Node \rightarrow Terminal\\
&Operator \rightarrow +|-|*|\div\\
&Terminal \rightarrow x_0|x_1|E\\
&E \rightarrow realNumber(represented\ by\ e)
\end{aligned}$$

An individual in the example problem would have two trees; and each of them would be allowed to have any of the functions in the function set, which for this example is $\{+, -, *, \div\}$, in their intermediate nodes; and any of $\{x_0, x_1, e\}$ in the leaves. This, for example, would be a legal individual:



4.2. Fitness evaluation

The fitness evaluation procedure is probably the most treated aspect of design in the literature when dealing with GP-based feature extraction. As has been stated before, the idea is to have the provided classifier's performance drive the evolution. To achieve that, GP-RFD calculates fitness in the following way:

1. Prerequisite: a previously built classifier (the one built from dataset A) needs to be provided. It is used as a black box.
2. Given an individual composed of a list of expression trees (one corresponding to each extracted attribute), a new dataset (dataset S) is built applying the transformations encoded in those expression trees to all the samples in dataset B.
3. The fitness of the individual is the classifier's accuracy on dataset S (training-set accuracy), calculated as the ratio of correctly classified samples over the total number of samples.

Fig. 1 presents a schematic representation of the procedure.

4.3. Genetic operators

This section details the choices made for selection, crossover and mutation operators. Since the objective of this work is not to squeeze the maximum possible performance from GP, but rather to show that it is an appropriate technique for the problem and that it can indeed solve it, we did not pay special attention to these choices, and picked the most common ones in the specialized literature.

- Tournament selection without replacement. To perform this selection, k individuals are first randomly picked from the population (where k is the tournament size), while avoiding using any member of the population more than once. The selected individual is then chosen as the one with the best fitness among those picked in the first stage.
- One-point crossover: for each dimension, a subtree from one of the parents is substituted by one from the other parent. The procedure is specified in Algorithm 1. An example, for one of the dimensions only, can be seen in Fig. 2.

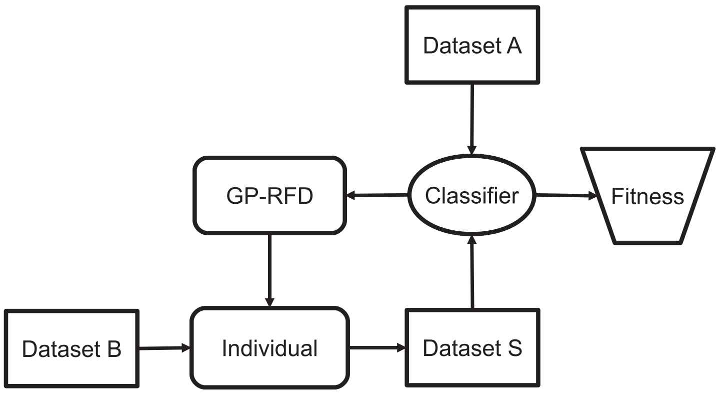


Fig. 1. Schematic representation of the fitness evaluation procedure.

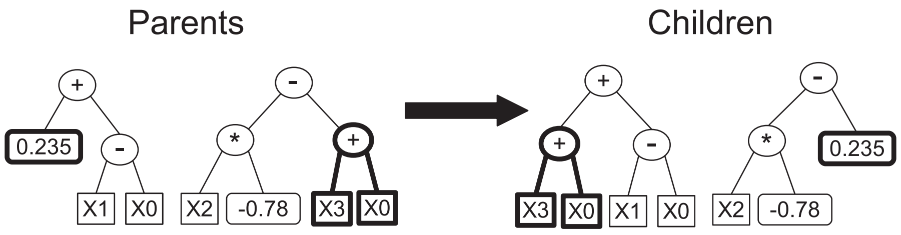


Fig. 2. Crossover example for one of the dimensions only, this is repeated for all dimensions (trees) on each individual.

- Swap mutation: this is a conservative mutation operator, that helps diversify the search within a close neighborhood of a given solution. It consists of exchanging the primitive associated to a node by one that has the same number of arguments.

Algorithm 1. One-point crossover procedure

```
FORALL trees on each individual
 1. Randomly select a non-root non-leave node on each of the two parents.
 2. The first child is the result of swapping the subtree below the selected node in the father for that of the mother.
 3. The second child is the result of swapping the subtree below the selected node in the mother for that of the father.
```

- Replacement mutation: this is a more aggressive mutation operator that leads to diversification in a larger neighborhood. The procedure to perform this mutation is the following:
  1. Randomly select a non-root non-leave node on the tree to mutate.
  2. Create a random tree of depth no more than a fixed maximum depth. This parameter has not been tinkered with, since the goal of this study was not to squeeze the maximum performance out of the proposed method, but rather to check its viability. Future work could tackle this issue.
  3. Swap the subtree below the selected node for the randomly generated one.

4.4. Parameters

The evolutionary parameters that were used for the experimental study are detailed in Table 1. As it was mentioned before, not much attention was payed to optimizing the parameters. Because of this the crossover and mutation probabilities, along with the number of generations to run, were fixed to the usual values in the literature (we could call them 'default values') and were not changed in any of the experiments.

Some of the evolutionary parameters are problem dependent, to select an appropriate value for them we used the following rules:

- Population size: since the only measure of difficulty we know about each of our problems a priori is the number of attributes present in the dataset (n_v), we have to fix the population size as a function of it. In the experiments carried out in this study, we found 400$*n_v$ to be a large enough population to achieve satisfactory results. This parameter is problem-dependent, so what we are fixing here is an upper bound for the population size needed. We found that, by following this

Table 1
Evolutionary parameters for a n_v-dimensional problem.

Parameter	Value
Number of trees	n_v
Population size	$400*n_v$
Duration of the run	50 generations
Selection operator	Tournament without replacement
Tournament size	$log_2(n_v)+1$
Crossover operator	One-point crossover
Crossover probability	0.9
Mutation operator	Replacement & Swap mutations
Replacement mutation probability	0.001
Swap mutation probability	0.01
Maximum depth of the swapped in subtree	5
Function set	Problem dependent
Terminal set	$\{x_0, x_1, \ldots, x_{n_v} - 1, e\}$

rule, GP-RFD consistently achieved good results; being able to solve the harder transformations, even though it was excessive for the easier ones and thus resulted in slower execution times. If harder problems than the ones studied in this paper were to be tackled, this parameter might need to be revised.
- Tournament size: since we are increasing the population size as a function of n_v, an increase of the selection pressure is needed too. The formula we used to calculate tournament size is: $log_2(n_v) + 1$. Again, this empirical estimation produced the best results; while an excessive pressure produced too fast of a convergence into local optima, and not enough pressure prevents GP-RFD from converging at all.

Table 2
Datasets used.

Dataset	Attributes	Samples	Classes	Class distribution	Attr. type
Linear synthetic	2	1000	2	50–50%	Real
Tao	2	1888	2	50–50%	Real
Iris	4	150	3	33–33–33%	Real
Phoneme	5	5404	2	70–30%	Real
Wisconsin	9	683	2	65–35%	Real
Heart	13	270	2	55–45%	Real
Wine	13	178	3	33–39%–27%	Real
Wdbc	30	569	2	65–45%	Real
Ionosphere	34	351	2	65–45%	Real
Sonar	60	208	2	54–46%	Real
Mux-11	11	2048	2	50–50%	Nominal
Cancer (A)	93	789	2	60–40%	Real
Cancer (B)	93	665	2	60–40%	Real

Table 3
Transformations performed on the Tao dataset.

Experiment	Rotation	Translate & extrude
Transformation applied	$b_0 = a_0{*}cos(1) + a_1{*}sin(1)$ $b_1 = a_0{*}sin(1) + a_1{*}cos(1)$	$b_0 = a_0{*}3 + 2$

Table 4
Transformations performed on the UCI and ELENA datasets.

Dataset	In-set transformation	Out-of-set transformation
Iris	$b_2 = a_2 + a_2$	$b_3 = e^{a_3}$
Phoneme	$b_0 = a_0 - 0.4$ $b_3 = a_3{*}2.5$	$b_0 = sin(a_0)$ $b_3 = cos(a_3)$
Wisconsin	$b_1 = a_1 + 2$ $b_5 = a_5{*}3$	$b_1 = cos(a_1)$ $b_5 = sin(a_5)$
Heart	$b_2 = a_2{*}2$ $b_11 = a_11 + 3$	$b_2 = sin(a_2)$ $b_11 = e^{a_11}$
Wine	$b_9 = a_9 - 1$ $b_12 = a_12{*}2$	$b_9 = sin(a_9)$ $b_12 = cos(a_12)$
Wdbc	$b_26 = a_26 - 1$ $b_27 = a_27{*}3$	$b_26 = sin(a_26)$ $b_27 = cos(a_27)$
Ionosphere	$b_4 = a_4 - 0.5$ $b_7 = a_7{*}2$	$b_4 = e^{a_4}$ $b_7 = sin(a_7)$
Sonar	$b_7 = a_7 + 0.3$ $b_43 = a_43{*}2$	$b_7 = sin(a_7)$ $b_43 = e^{a_43}$

Table 5
Transformations performed on the Multiplexer-11 dataset.

Experiment	Bit flip	Column swap
Transformation applied	$b_1 = not(a_1)$	$b_1 = a_2$ $b_2 = a_3$ $b_3 = a_1$

Table 6
Experimental parameters.

Dataset	Population size	Tournament size	Function set
Linear synthetic	800	2	$\{+,-,*,\div\}$
Tao	800	2	$\{+,-,*,\div\}$
Iris	1600	3	$\{+,-,*,\div\}$
Phoneme	2000	3	$\{+,-,*,\div\}$
Wisconsin	3600	4	$\{+,-,*,\div\}$
Heart	5200	4	$\{+,-,*,\div\}$
Wine	5200	4	$\{+,-,*,\div\}$
Wdbc	12,000	5	$\{+,-,*,\div\}$
Ionosphere	13,600	6	$\{+,-,*,\div\}$
Sonar	24,000	6	$\{+,-,*,\div\}$
Mux-11	4400	4	$\{+,-,*,\div\}$
Cancer	37,200	6	$\{+,-,*,\div,exp,cos\}$

4.5. Function set

Which functions to include in the function set are usually dependent on the problem. , , , Since one of our goals is to have an algorithm as universal and robust as possible, where the user does not need to fine-tune any parameters to achieve good performance; we decided not to study the effect of different function set choices. The used function sets are chosen to be close to the default ones most authors use in the literature, and were extracted in all cases from $\{+,-,*,\div,exp,cos\}$. The benchmark experiments did not use $\{exp,cos\}$, since we intended to test the capability of the method to unveil transformations that did not include functions in the function set. The specific choices for each of the experiments can be seen in Table 6.

4.6. Execution flow

Algorithm 2 contains a summary of the execution flow of the GP procedure, which follows a classical evolutionary scheme. It stops after a user-defined number of generations, providing as a result the best individual (i.e., transformation) it has ever found.

Algorithm 2. Execution flow of the GP procedure

```
1. Randomly create the initial population by applying the context-free grammar presented in Subsection 4.1.
2. Repeat Ng times (where $Ng$ is the number of generations)
   2.1 Evaluate the current population, using the procedure shown in Subsection 4.2.
   2.2 Apply selection and crossover to create a new population that will replace the old one.
   2.3 Apply the mutation operators to the new population.
3. Return the best individual ever seen.
```

5. Experimental study

This section is organized in the following way: to begin with, a general description of the experimental procedure is presented in Subsection 5.1, along with the datasets that we have used for our testing (both the benchmark problems and the prostate cancer dataset); and also in the benchmarks' case the transformations performed on each of them. The parameters used for each experiment are shown in Subsection 5.2; followed by a presentation of the benchmark experimental results in Subsection 5.3. Finally, the results obtained on the prostate cancer problem are presented in Subsection 5.4.

5.1. Experimental framework, datasets and transformations

The goal of the experiments was to check how effective GP-RFD was in finding a transformation over dataset B that would increase the provided classifier's accuracy. To validate our results, we employed a 5-fold cross validation technique [31]. We used the beagle library [18] for our GP implementation.

The experimental study is fractioned in two parts. In the first one, a synthetic set of tests is built from a few well-known benchmark datasets. The procedure followed in these experiments was (see Fig. 3 for a schematic representation):

1. Split the original dataset in two halves with equal class distribution.
2. Consider the first half, to be dataset A.
3. From dataset A, build a classifier. We chose C4.5 [50], but any other classifier would work exactly the same; due to the fact that GP-RFD uses the learned classifier as a black box.

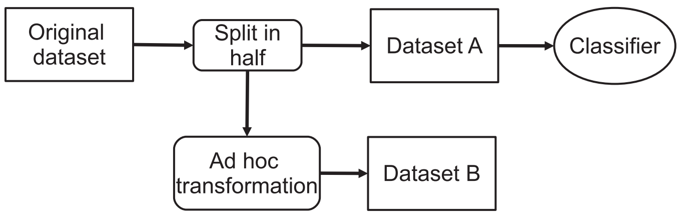


Fig. 3. Schematic representation of the experimental procedure with benchmark datasets.

4. Apply a transformation over the second half of the original dataset, creating dataset B. The transformations we tested were designed to check GP-RFD's performance on different types of problems, including both linear and non-linear transformations. A description of each of them can be found in the next subsection.
5. The performance of the classifier built in step 2 is significantly worse on dataset B than it is on dataset A. This is the starting point on the real problem we are emulating.
6. Apply GP-RFD to dataset B in order to evolve a transformation that will create a solution dataset S. Use 5-fold cross validation over dataset S, so that training and test set accuracy results can be obtained.
7. Check the performance of the step 2 classifier on dataset S. Ideally, it should be close to the one on dataset A, which would mean GP-RFD has successfully discovered the hidden transformation and inverted.

The second part of the study is the application of the proposed algorithm to the prostate cancer problem. The steps followed in this case were:

1. Consider each of the provided datasets to be datasets A and B respectively.
2. From dataset A, build a classifier. Use 5-fold cross validation to obtain training and test-set performance results.
3. Apply GP-RFD to dataset B in order to evolve a transformation that will create a solution dataset S. Use 5-fold cross validation over dataset S, so that training and test set accuracy results can be obtained.
4. Check the performance of the step 2 classifier on dataset S. Ideally, it should be close to the one on dataset A, meaning GP-RFD has successfully discovered the hidden transformation and inverted it.

The selected datasets are summarized in Table 2. A short description and motivation for each of the datasets follows, and this subsection is concluded with the specification of the transformations that were fabricated to test the algorithm on each of the benchmark datasets. For the two-dimensional problems, the transformations are also graphically represented.

Note that the transformations in the prostate cancer problem are not specified. This is due to it being a real-world problem and not a fabricated one, so the implicit transformations in the data were unknown a priori.

- Linear synthetic dataset: we have called the first dataset 'Linear synthetic'. It was created specifically for this work, with the idea of having an easily representable linearly separable dataset to work with. It was chosen to check the performance of GP-RFD on some simple transformations, without the added difficulty of having a complex original dataset. The dataset can be seen in Fig. 4. We applied three transformations to this dataset A: rotation, translation and extrusion and circle. The transformed datasets (datasets B on the experiments) can be seen in Figs. 5–7 respectively.

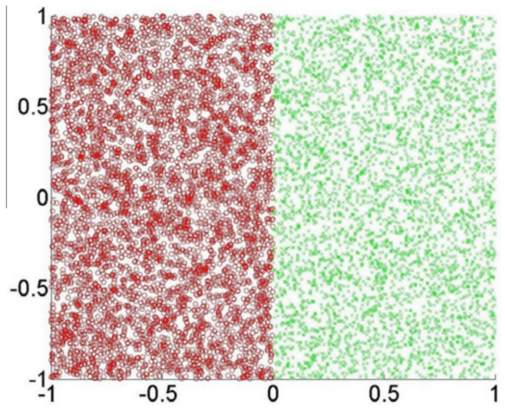


Fig. 4. Linear synthetic dataset, dataset A.

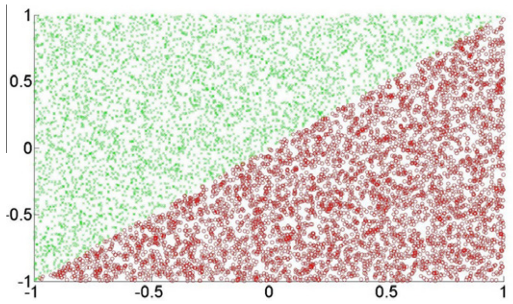


Fig. 5. Rotation problem, transformed dataset.

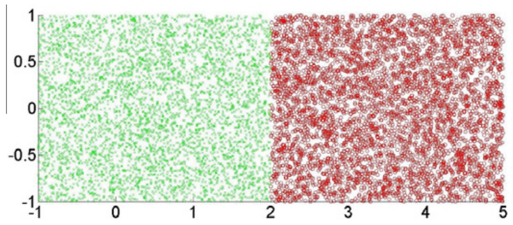


Fig. 6. Translation & extrusion problem, transformed dataset.

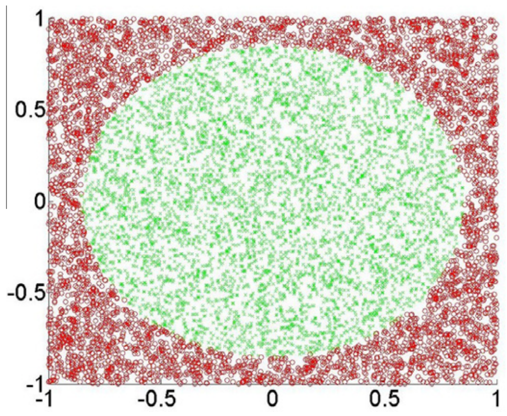


Fig. 7. Circle problem, transformed dataset.

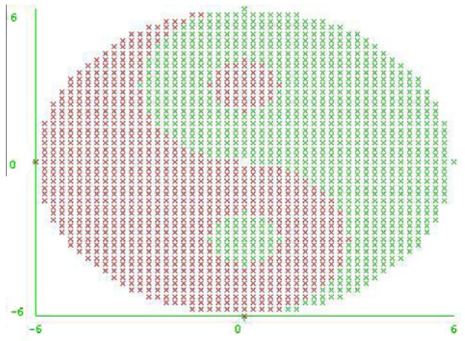


Fig. 8. Tao dataset. This is dataset A, over which the different transformations are applied, and the transformed datasets have to fit to the same classifier this dataset does.

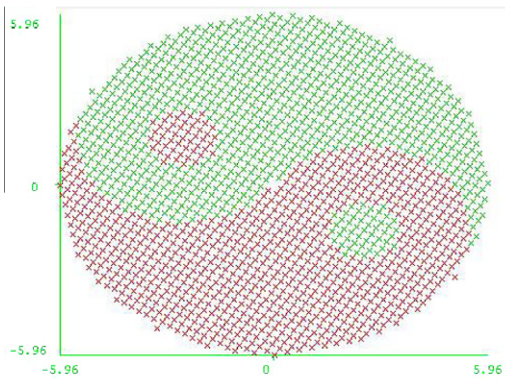


Fig. 9. Rotated Tao, transformed dataset.

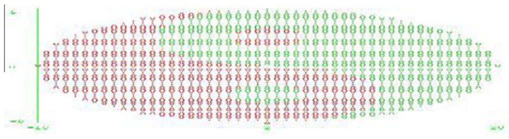


Fig. 10. Translated and extruded Tao, transformed dataset.

- Tao: the next step to check the usefulness of GP-RFD is starting from a harder dataset. To this end, we chose the Tao dataset, still a 2-dimensional problem but where classification is much harder. This dataset is also built artificially [41]. The dataset can be seen, before any transformations (dataset A), in Fig. 8. Mirroring the transformations applied over the linear synthetic dataset, we chose to transform the original Tao dataset by rotating it (Fig. 9); or by translating and extruding (Fig. 10). The transformations applied to Tao can also be seen in Table 3.
- UCI and ELENA datasets: once GP-RFD has been tested in small (with a low number of attributes) datasets, it is useful to see how it fares in bigger benchmark problems. We chose a few different datasets from the UCI database [4], as well as the ELENA project [23]:
  - Iris: classification of iris plants (UCI).
  - Phoneme: distinguish between nasal and oral sounds (ELENA).
  - Wisconsin: diagnosis of breast cancer patients (UCI).
  - Heart: detect the absence or presence of heart disease (UCI).
  - Wine: classification of different types of Italian wines (UCI).
  - Wdbc: determination of whether a found tumor is benign or malignant (UCI).
  - Ionosphere: radar data where the task is to decide is a given radar return is good or bad (UCI, modified as found in the KEEL database [1]).
  - Sonar: distinguishing between rocks and metal cylinders from sonar data (UCI).

We performed two different experiments on each of the datasets. In the first experiment, the transformation is created using functions that appear in the function set of the GP procedure (more specifically, one of the attributes is added to itself). We named this experiment 'in-set transformation'. The second one transforms the dataset by using functions that do not appear in the GP function set. The name for this experiment is 'out-of-set transformation'. The exact details for these transformations can be found in Table 4. Any attribute not specified as being part of the transformation in the tables is assumed to be unchanged.

- Multiplexer-11: since GP-RFD should be flexible enough to be able to tackle datasets with nominal attributes, one of these datasets was included in the testing. In this work, we chose the Multiplexer problem. This is a binary problem where some of the bits act as address, and the remaining bits are data registers. The correct classification for a given input is the value of the register pointed by the address bits. The specific instance used here is Multiplexer-11, a dataset with 11 binary attributes (where the first three act as address, and the remaining eight as registers); and $2^{11} = 2048$ samples. Two different transformations were tested: in the first one, of the address bits was flipped; while in the second experiment there was an attribute swap, in a circular shift. The details can be found in Table 5.
- Prostate cancer: as was explained in Section 3, the solution to this problem is the main motivation for this work. Since we were provided with data from two real laboratories, there was no need to fabricate any transformations: we chose one the data from one of the laboratories as dataset A and the other one as dataset B.

5.2. Parameters

In this section, we detail the parameters used for each of the datasets, including both the evolutionary parameters and the GP setup. The parameters were chosen following the rules detailed in Section 4.4.

As can be seen in Table 6, the population sizes are large. This is mostly due to GP being a technique that traditionally requires large population sizes to be effective, a factor which is aggravated by the fact that GP-RFD evolves multiple expression trees simultaneously (one for each attribute in the dataset). We acknowledge this issue provokes long execution times for some of the experiments, but considered it a secondary concern and did not address it in this work.

5.3. Experimental results: benchmark problems

This part presents the results obtained in terms of classifier performance for the benchmark problems, along with a statistical analysis to evaluate whether GP-RFD is effective.

Table 7 details the performance obtained by the C4.5 classifier on each of the benchmark problems. It includes the classifier performance, calculated as shown on Subsection 4.2, on:

- Dataset A, which was used to generate the decision tree. A 5-fold cross validation technique was applied, and both training and test set results are presented.
- Dataset B, which was created by designing an ad hoc transformation.
- Dataset S, which is the result of applying GP-RFD to dataset B, obtaining a transformed dataset where classifier performance is increased. A 5-fold cross validation technique was applied, and both training and test set results are presented.

The results show that GP-RFD is capable of reversing nearly all of the fabricated transformations, achieving accuracy rates that are very close to the ones obtained in the original datasets in both training and test performances. GP-RFD has also proven capable of generalizing well, as can be seen by the small difference between training and test set classification performances in most cases. However, some of the datasets (which, coincidentally, tend to also behave badly in terms of generalization when building classifiers) present some generalization issues, leading to the inability to fully solve the problem dataset.

5.3.1. Statistical analysis

To complete the experimental study, we have performed a statistical comparison between the classifier performance over the following datasets:

- Dataset A, from which the classifier was built.
- Dataset B, artificially built by injecting an ad hoc transformation.

Table 7
Classifier performance results: benchmark problems.

Problem	Classifier performance on dataset ...				
	A-training	A-test	B	S-training	S-test
Linear synthetic – rotation	1.00000	1.00000	0.24930	1.00000	1.00000
Linear synthetic – translation& extrusion	1.00000	1.00000	0.34160	1.00000	0.99800
Linear synthetic – circle	1.00000	1.00000	0.49860	0.96050	0.94400
Tao – rotation	0.98518	0.93750	0.62924	0.94418	0.94255
Tao – translation& extrusion	0.98518	0.93750	0.80403	0.95344	0.93192
Iris – in-set functions	0.97330	0.93333	0.66667	0.99333	0.92000
Iris – out-of-set functions	0.97330	0.93333	0.60000	0.99000	0.92000
Phoneme – in-set functions	0.91895	0.84160	0.75204	0.828978	0.769907
Phoneme – out-of-set functions	0.91895	0.84160	0.59141	0.839871	0.804815
Wisconsin – in-set functions	0.97361	0.93842	0.35380	0.98248	0.93821
Wisconsin – out-of-set functions	0.97361	0.93842	0.88889	0.98321	0.94412
Heart – in-set functions	0.89630	0.72593	0.45296	0.92778	0.79259
Heart – out-of-set functions	0.89630	0.72593	0.60000	0.96296	0.72594
Wine – in-set functions	0.97727	0.89733	0.65556	0.98889	0.90000
Wine – out-of-set functions	0.97727	0.89733	0.40000	0.96944	0.91111
Wdbc – in-set functions	0.98571	0.92143	0.57143	0.98839	0.946428
Wdbc – out-of-set functions	0.98571	0.92143	0.82857	0.98214	0.97500
Ionosphere – in-set functions	0.98286	0.87429	0.70857	0.98571	0.88571
Ionosphere – out-of-set functions	0.98286	0.87429	0.77714	0.98571	0.857143
Sonar – in-set functions	0.93939	0.60601	0.61000	0.95500	0.66000
Sonar – out-of-set functions	0.93939	0.60601	0.51000	0.94750	0.72000
Mux11 – bit flip	1.00000	0.97070	0.50000	0.96951	0.96667
Mux11 – column swap	1.00000	0.97070	0.62500	0.97195	0.96765

- Dataset S-test, the result of applying GP-RFD over dataset B (test-set results).

In [13,21,19,20] a set of simple, safe and robust non-parametric tests for statistical comparisons of classifiers are recommended. One of them is the Wilcoxon signed-ranks test [57,53], which is the test that we have selected to do the comparison.

This is analogous to the paired t-test in non-parametric statistical procedures; therefore it is a pairwise test that aims to detect significant differences between two sample means, that is, the behavior of two algorithms. It is defined as follows: let d_i be the difference between the performance scores of the two classifiers on the *ith* dataset out of N_{ds} datasets. The differences are ranked according to their absolute values; average ranks are assigned in the case of ties. Let R^+ be the sum of ranks for the data-sets in which the first algorithm outperformed the second, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$\begin{aligned} R^+ &= \sum_{d_i>0} ank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \\ R^- &= \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \end{aligned} \tag{1}$$

Let T be the smaller of the sums, $T = min(R^+, R^-)$. If T is less than or equal to the value of the distribution of Wilcoxon for N_{ds} degrees of freedom [61], the null hypothesis of equality of means is rejected; this will mean that a given classifier outperforms their opposite, with the p-value associated.

The Wilcoxon signed-ranks test is more sensitive than the t-test. It assumes commensurability of differences, but only qualitatively: greater differences still count for more, which is probably desired, but the absolute magnitudes are ignored. From a statistical point of view, the test is safer since it does not assume normal distributions. Also, outliers (extremely good/bad performances) have a smaller effect on the Wilcoxon signed-ranks test than on the t-test.

When the assumptions of the paired t-test are met, the Wilcoxon signed-ranks test is less powerful than the paired t-test. On the other hand, when the assumptions are not met, the Wilcoxon test is a better choice than the t-test. This is because the Wilcoxon test can be applied over the averaged results obtained by the algorithms in each data set, without any assumptions about the characteristics of the distribution of the results obtained.

A complete description of the Wilcoxon signed ranks test and other non-parametric tests for pairwise and multiple comparisons, together with software for their use, can be found in the website available at http://sci2s.ugr.es/sicidm/.

As it was mentioned above, the test was applied to compare the classifier performance in datasets A, B and S. The results can be seen in Table 8. Note that we compare the results in dataset A against those in S both in terms of training and test sets. However, since the classifier was not built from dataset B, we consider those results test-set related and compare it with S-test.

So we can conclude GP-RFD is capable of finding transformations resulting in a new dataset S that

1. Significantly outperforms dataset B in terms of classifier performance.
2. Obtains statistically equivalent results to dataset A, both in terms of training and test sets. Since the classifier was built from dataset A, this means dataset S is a successful repair of the fracture between datasets A and B, assuming class dis-

Table 8
Wilcoxon signed-ranks test results: Benchmark problems.

Comparison	R^+	R^-	*p*-Value	Null hypothesis of equality
A-test vs B	275	1	4.77E−007	*rejected* (A-test outperforms B)
B vs S-test	0	276	2.38E−007	*rejected* (S-test outperforms B)
A-training vs S-training	147.5	128.5	–	*accepted*
A-test vs S-test	128.5	147.5	–	*accepted*

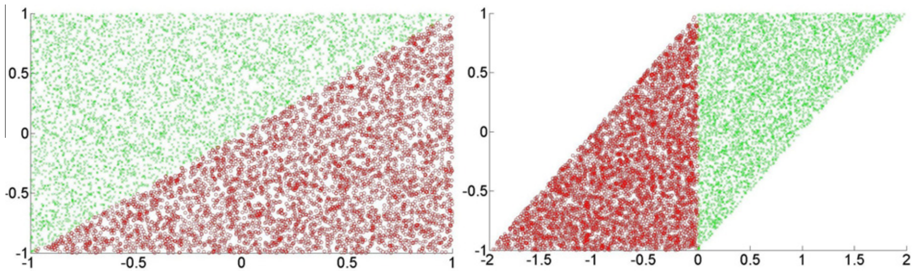


Fig. 11. Linear synthetic rotation, problem (L) and solution (R) datasets.

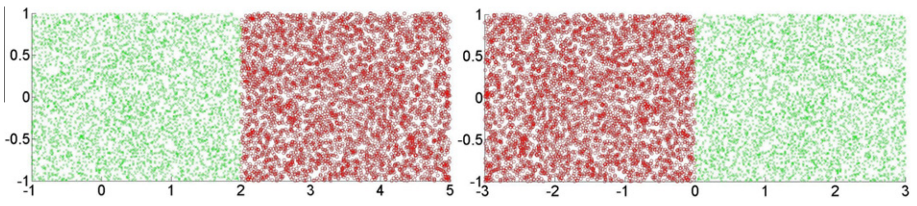


Fig. 12. Linear synthetic translation and extrusion, problem (L) and solution (R) datasets.

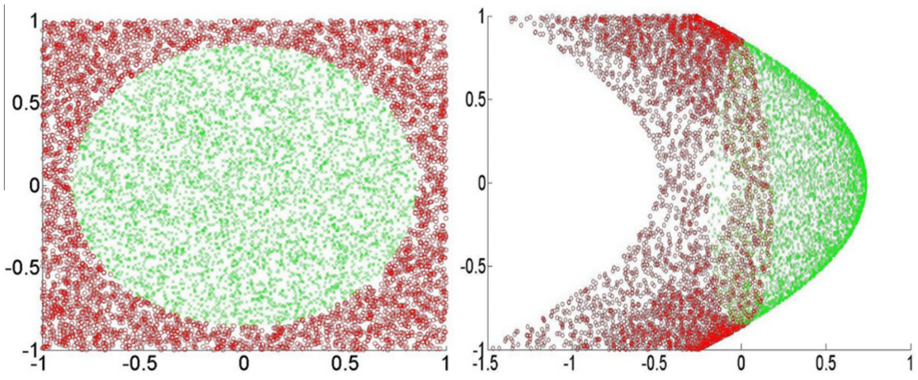


Fig. 13. Circle, problem (L) and solution (R) datasets.

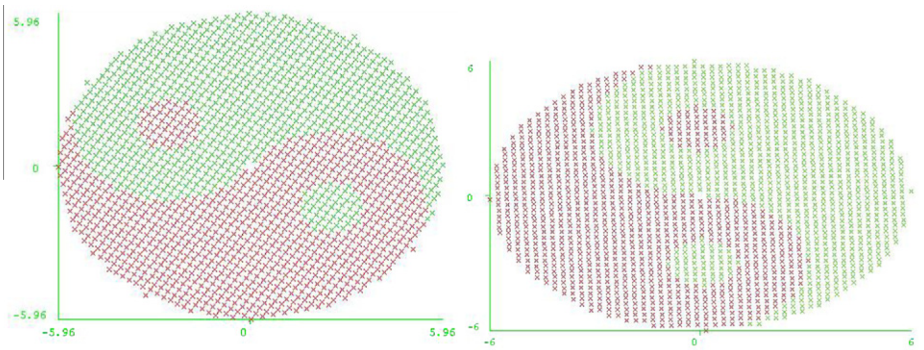


Fig. 14. Rotation in Tao, problem (L) and solution (R) datasets.

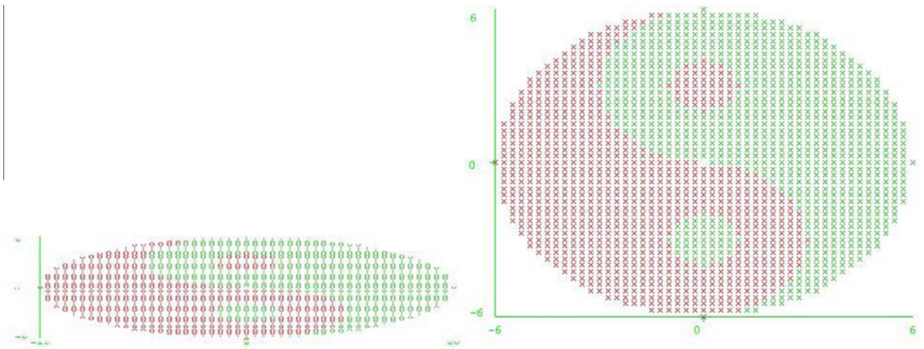


Fig. 15. Translation and extrusion in Tao, problem (L) and solution (R) datasets.

tribution did not change. We know this is the case in these experiments due to the way we built datasets A and B, but it has to be kept in mind when applying the method in other environments.

5.3.2. Graphical results

This section presents graphical representations of some of the obtained results. Since several of the datasets have a high number of variables that make them extremely hard to chart in a simple way, only the results corresponding to the linear synthetic dataset (Figs. 11–13) and the Tao dataset (Figs. 14 and 15) are shown. To make the visualization easier, each of the solution datasets (datasets S) is presented side-by-side with the corresponding problem dataset (datasets B). The original datasets (datasets A) can be seen in Fig. 4 for the linear synthetic dataset and Fig. 8 for the Tao dataset.

5.4. Prostate cancer experimental results

This section presents the preliminary results for the Prostate Cancer problem, in terms of classifier accuracy. The results obtained can be seen in Table 9. In that table, dataset A is the one from the first lab; which was used to build the classifier, dataset B is the one coming from the second lab, and dataset S is the result of the application of GP-RFD.

To check whether the full dataset B was needed to evolve an effective transformation, we also tested using just half of it to train GP-RFD, and the other half to test (2-fold cross validation). These results are also included in Table 9.

The performance results are excellent for a number of reasons. First and foremost, GP-RFD was able to find a transformation over the data from the second laboratory that made the classifier work just as well as it did on the data from the first lab, effectively finding the hidden perturbations that prevented the classifier from working accurately.

The second positive conclusion to be obtained from the results is the generalization power of GP-RFD. As can be observed from the test results, GP-RFD does not ‘cheat’ by over-learning on the known data, and works well when transforming new, previously unseen, samples.

Third, the results show GP-RFD was capable of obtaining excellent results using just half of the B dataset to train. This result highlights the power of the method to unveil the hidden transformation from a relatively low number of samples.

We also performed a Wilcoxon signed-ranks test to evaluate the performance of GP-RFD over the case of study problem. In order to do it, we used the results from each partition in the 5-fold cross validation procedure. We ran the experiment four times, resulting in $4 * 5 = 20$ performance samples to carry out the statistical test. As we did before, R^+ corresponds to the first algorithm in the comparison winning, and R^- to the second one. Table 10 shows the results.

The results on the case study problem are exactly the same as those achieved in the benchmark problems. We can then conclude GP-RFD was capable of repairing the existing fracture between the data from both laboratories. Again, this conclusion assumes class distribution did not change. It is a given in this case, since we know the class distribution to be equal in datasets A and B, but is an issue that has to be kept in mind when applying the method to other problems.

6. Concluding remarks

We have presented GP-RFD, a new algorithm that approaches a common problem in real life for which not many solutions have been proposed in evolutionary computing. The problem in question is the repairing of fractures between data by adjusting the data itself, not the classifiers built from it.

We have developed a solution to the problem by means of a GP-based algorithm that performs feature extraction on the problem dataset driven by the accuracy of the previously built classifier.

Table 9
Classifier performance results: the prostate cancer problem.

Validation method	Classifier performance in dataset ...				
	A-training	A-test	B	S-training	S-test
5-fold cross validation	0.95435	0.92015	0.83570	0.95191	0.92866
2-fold cross validation	0.95435	0.92015	0.83570	0.95482	0.93223

Table 10
Wilcoxon signed-ranks test results: the prostate cancer problem.

Comparison	R^+	R^-	*p*-Value	Null hypothesis of equality
A-test vs B	210	0	1.91E−007	*rejected* (A-test outperforms B)
B vs S-test	0	210	1.91E−007	*rejected* (S-test outperforms B)
A-training vs S-training	126	84	–	*accepted*
A-test vs S-test	84	126	–	*accepted*

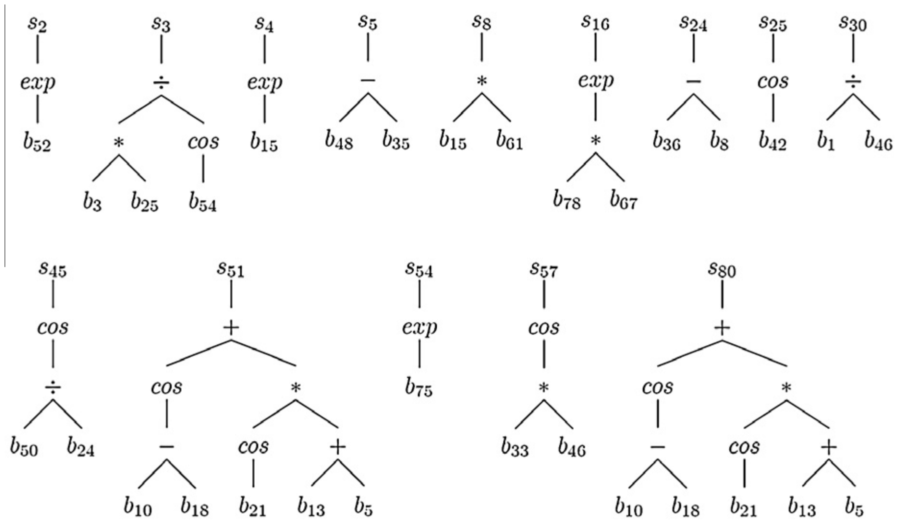


Fig. 16. Tree representation of the expressions contained in a solution to the prostate cancer problem.

We have tested GP-RFD on a set of artificial benchmark problems, where a problem dataset is fabricated by applying an ad hoc disruption to an original dataset, and it has proved capable of solving all the transformations presented showing good performance both in train and, more importantly, test data.

We have also being able to apply GP-RFD to a real-world problem where data from two different laboratories regarding prostate cancer diagnosis was provided, and where the classifier learned from one did not perform well enough on the other. Our algorithm was capable of learning a transformation over the second dataset that made the classifier fit just as well as it did on the first one. The validation results with 5-fold cross validation also support the idea that the algorithm is obtaining good results; and has a strong generalization power.

Lastly, we have applied a statistical analysis methodology that supports the claim that the classifier performance obtained on the solution dataset significantly outperforms the one obtained on the problem dataset.

There is, however, one point where the proposed method has not been successful. The learned transformations have failed to provide any information about why the fracture appeared between the data from the two laboratories. We have, however, included a sample of the transformations learned in appendix A.

Acknowledgments

Jose García Moreno-Torres was supported by a scholarship from 'Obra Social la Caixa' and is currently supported by a FPU grant from the Ministerio de Educación y Ciencia of the Spanish Government, and also by the KEEL project (TIN2008-06681-C06-01). Rohit Bhargava would like to acknowledge collaborators over the years, especially Dr. Stephen M. Hewitt and Dr. Ira W. Levin of the National Institutes of Health, for numerous useful discussions and guidance. Funding for this work was provided in part by University of Illinois Research Board and by the Department of Defense Prostate Cancer Research Program. This work was also funded in part by the National Center for Supercomputing Applications and the University of Illinois, under the auspices of the NCSA/UIUC faculty fellows program.

Appendix A. Sample solution from the prostate cancer problem

In this appendix, we include a sample of the learned transformations for the prostate cancer problem, presenting the transformations corresponding to the highest fitness individual ever found. Due to space concerns, only the attributes relevant to the C4.5 classifier are shown (Fig. 16).

References

[1] J. Alcalá-fdez, L. Sánchez, S. García, M.J.D. Jesus, S. Ventura, J.M. Garrell, J. Otero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, Keel: a software tool to assess evolutionary algorithms for data mining problems, Soft Computing - A Fusion of Foundations, Methodologies and Applications 13 (3) (2008) 307–318.

[2] AmericanCancerSociety. How many men get prostate cancer? <http://www.cancer.org/docroot/CRI/content/CRI_2_2_1X_How_many_men_get_prostate_cancer_36.asp>.

[3] A. Arcuri, X. Yao, Co-evolutionary automatic programming for software development, Information Sciences (2010), in press, http://dx.doi.org/10.1016/j.ins.2009.12.019.

[4] A. Asuncion, D. Newman, UCI machine learning repository (2007).

[5] W. Banzhaf, F.D. Francone, P. Nordin, The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets, in: In Parallel Problem Solving from Nature IV, Proceedings of the International Conference on Evolutionary Computation, Springer Verlag, 1996, pp. 300–309.
[6] F. Berlanga, A. Rivera, M. del Jesus, F. Herrera, GP-COACH: genetic programming-based learning of compact and accurate fuzzy rule-based classification systems for high-dimensional problems, Information Sciences 180 (8) (2010) 1183–1200.
[7] A.S. Bickel, R.W. Bickel, Tree structured rules in genetic algorithms, In Proceedings of the Second International Conference on Genetic Algorithms and their Application, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1987.
[8] M.C.J. Bot, Feature extraction for the k-nearest neighbour classifier with genetic programming, In EuroGP '01: Proceedings of the Fourth European Conference on Genetic Programming, Springer-Verlag, London, UK, 2001.
[9] D.A. Cieslak, N.V. Chawla, A framework for monitoring classifiers' performance: when and why failure occurs?, Knowledge and Information Systems 18 (1) (2009) 83–108
[10] N.L. Cramer, A representation for the adaptive generation of simple sequential programs, In Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1985.
[11] C. Darwin, On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life, John Murray, London, UK, 1859.
[12] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[13] P.G. Espejo, S. Ventura, F. Herrera, A survey on the application of genetic programming to classification, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40 (2) (2010) 121–144.
[14] M. Evett, T. Fernandez, Numeric mutation improves the discovery of numeric constants in genetic programming, in: J. Koza (Ed.), Proceedings of the Third Annual Genetic Programming Conference, Morgan Kaufmann, Madison, WI, 1998, pp. 66–71.
[15] D.C. Fernandez, R. Bhargava, S.M. Hewitt, I.W. Levin, Infrared spectroscopic imaging for histopathologic recognition, Nature Biotechnology 23 (4) (2005) 469–474.
[16] C. Fujiko, J. Dickinson, Using the genetic algorithm to generate lisp source code to solve the prisoner's dilemma, in: Proceedings of the Second International Conference on Genetic Algorithms and their application, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1987, pp. 236–240.
[17] C. Gagné, M. Parizeau, Genericity in evolutionary computation software tools: principles and case study, International Journal on Artificial Intelligence Tools 15 (2) (2006) 173–194.
[18] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Computing 13 (10) (2009) 959–977.
[19] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Information Sciences 180 (10) (2010) 2044–2064.
[20] S. García, F. Herrera, An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[21] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[22] A. Guérin-Dugué et al. Deliverable R3-B1-P - Task B1: Databases. Technical report, Elena-NervesII "Enhanced Learning for Evolutive Neural Architecture, ESPRIT-Basic Research Project Number 6891, June 1995, Anonymous FTP:/pub/neural-nets/ELENA/Databases.ps.Z on ftp.dice.ucl.ac.be.
[23] H. Guo, A.K. Nandi, Breast cancer diagnosis using genetic programming generated feature, Pattern Recognition 39 (5) (2006) 980–987.
[24] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.
[25] I. Guyon, S. Gunn, M. Nikravesh, L. Zadeh (Eds.), Feature Extraction, Foundations and Applications, Springer, 2006.
[26] C. Harris, An investigation into the Application of Genetic Programming techniques to Signal Analysis and Feature Detection, PhD thesis, University College, London, 26 Sept. 1997.
[27] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, USA, 1975.
[28] K.-A. Kim, S.-Y. Oh, H.-C. Choi, Facial feature extraction using pca and wavelet multi-resolution images, in: Sixth IEEE International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society, Los Alamitos, CA, USA, 2004, p. 439.
[29] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1995, pp. 1137–1143.
[30] M. Kotani, S. Ozawa, M. Nakai, K. Akazawa, Emergence of feature extraction function using genetic programming, In KES (1999) 49–152.
[31] P. Kouchakpour, A. Zaknich, T. Bräunl, Population variation in genetic programming, Information Sciences 177 (17) (2007) 3438–3452.
[32] P. Kouchakpour, A. Zaknich, T. Bräunl, Dynamic population variation in genetic programming, Information Sciences 179 (8) (2009) 1078–1091.
[33] J. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press, Cambridge, MA, 1992.
[34] J. Koza, Genetic programming II: Automatic Discovery of Reusable Programs, Complex Adaptive Systems, MIT Press, Cambridge, Mass, 1994.
[35] J.R. Koza, M.J. Streeter, M.A. Keane, Routine high-return human-competitive automated problem-solving by means of genetic programming, Information Sciences 178 (23) (2008) 4434–4452.
[36] I.W. Levin, R. Bhargava, Fourier transform infrared vibrational spectroscopic imaging: integrating microscopy and molecular recognition, Annual Review of Physical Chemistry 56 (2005) 429–474.
[37] J.-Y. Lin, H.-R. Ke, B.-C. Chien, W.-P. Yang, Classifier design with feature selection and feature extraction using layered genetic programming, Expert Systems with Applications 34 (2) (2008) 1384–1393.
[38] H. Liu, H. Motoda, Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic, Boston, 1998. vol. SECS 453.
[39] X. Llorà, J.M. Garrell, Knowledge-independent data mining with fine-grained parallel evolutionary algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), Morgan Kaufmann Publishers, 2001, pp. 461–468.
[40] X. Llorà, A. Priya, R. Bhargava, Observer-invariant histopathology using genetics-based machine learning, Natural Computing: An International Journal 8 (1) (2009) 101–120.
[41] X. Llorà, R. Reddy, B. Matesic, R. Bhargava, Towards better than human capability in diagnosing prostate cancer using infrared spectroscopic imaging, in: GECCO '07: Proceedings of the Ninth Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, 2007, pp. 2098–2105.
[42] U.-M. O'Reilly, An Analysis of Genetic Programming, PhD thesis, Carleton University, Ottawa-Carleton Institute for Computer Science, Ottawa, Ontario, Canada, 1995.
[43] M. Pei, E.D. Goodman, W.F. Punch. Pattern discovery from data using genetic algorithms, in: Proceeding of First Pacific-Asia Conference Knowledge Discovery & Data Mining(PAKDD-97), 1997.
[44] A. Piszcz, T. Soule, A survey of mutation techniques in genetic programming, in: GECCO '06: Proceedings of the Eigth Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, 2006, pp. 951–952.
[45] I.T. Podolak, Facial component extraction and face recognition with support vector machines, in: FGR '02: Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, IEEE Computer Society, Washington, DC, USA, 2002, p. 83.
[46] R. Poli, W.B. Langdon, N.F. Mcphee, A Field Guide to Genetic Programming, Lulu Enterprises Ltd, UK, 2008.
[47] J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, N.D. Lawrence, Dataset Shift in Machine Learning, The MIT Press, 2009.
[48] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
[49] C. Ryan, M. Keijzer, An analysis of diversity of constants of genetic programming, in: C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, E. Costa (Eds.), Genetic Programming, Proceedings of EuroGP'2003, LNCS, vol. 2610, Springer-Verlag, Essex, 2003, pp. 404–413.
[50] J.R. Sherrah, R.E. Bogner, A. Bouzerdoum, The evolutionary pre-processor: automatic feature extraction for supervised classification using genetic programming, Proceedings of the Second International Conference on Genetic Programming, vol. GP-97, Morgan Kaufmann, 1997, pp. 304–312.
[51] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed., Chapman & Hall/CRC, 2007.

[52] M.G. Smith, L. Bull, Genetic programming with a genetic algorithm for feature construction and selection, Genetic Programming and Evolvable Machines 6 (3) (2005) 265–281.
[53] W.A. Tackett, Genetic programming for feature discovery and image discrimination, in: Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993, pp. 303–311.
[54] K. Wang, S. Zhou, C.A. Fu, J.X. Yu, F. Jeffrey, X. Yu, Mining changes of classification by correspondence tracing, in: Proceedings of the 2003 SIAM International Conference on Data Mining (SDM 2003), 2003.
[55] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.
[56] N. Wyse, R. Dubes, A. Jain, A critical evaluation of intrinsic dimensionality algorithms a critical evaluation of intrinsic dimensionality algorithms, in: E.S. Gelsema, L.N. Kanal (Eds.), Pattern Recognition in Practice, Morgan Kauffman Publishers, Inc., Amsterdam, 1980, pp. 415–425.
[57] Y. Yang, X. Wu, X. Zhu, Conceptual equivalence for contrast mining in classification learning, Data & Knowledge Engineering 67 (3) (2008) 413–429.
[58] A. Zafra, S. Ventura, G3P-MI: a genetic programming algorithm for multiple instance learning, Information Sciences, 180 (23) (2010) 4496–4513.
[59] J.H. Zar, Biostatistical Analysis, 5th ed., Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2007.
[60] Y. Zhang, H. Li, M. Niranjan, P. Rockett, Applying cost-sensitive multiobjective genetic programming to feature extraction for spam E-mail filtering, in: Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008, Lecture Notes in Computer Science, vol. 4971, Springer, Naples, 2008, pp. 325–336.
[61] Y. Zhang, P.I. Rockett, A generic optimal feature extraction method using multiobjective genetic programming, Technical Report VIE 2006/001, Department of Electronic and Electrical Engineering, University of Sheffield, UK, 2006.
[62] Y. Zhang, P.I. Rockett, A generic multi-dimensional feature extraction method using multiobjective genetic programming, Evolutionary Computation 17 (1) (2009) 89–115.